

msdn magazine



Improvements
in ASP.NET Core 2.1.....20



Best-of-Breed UI Components for the Desktop, Web and Your Mobile World

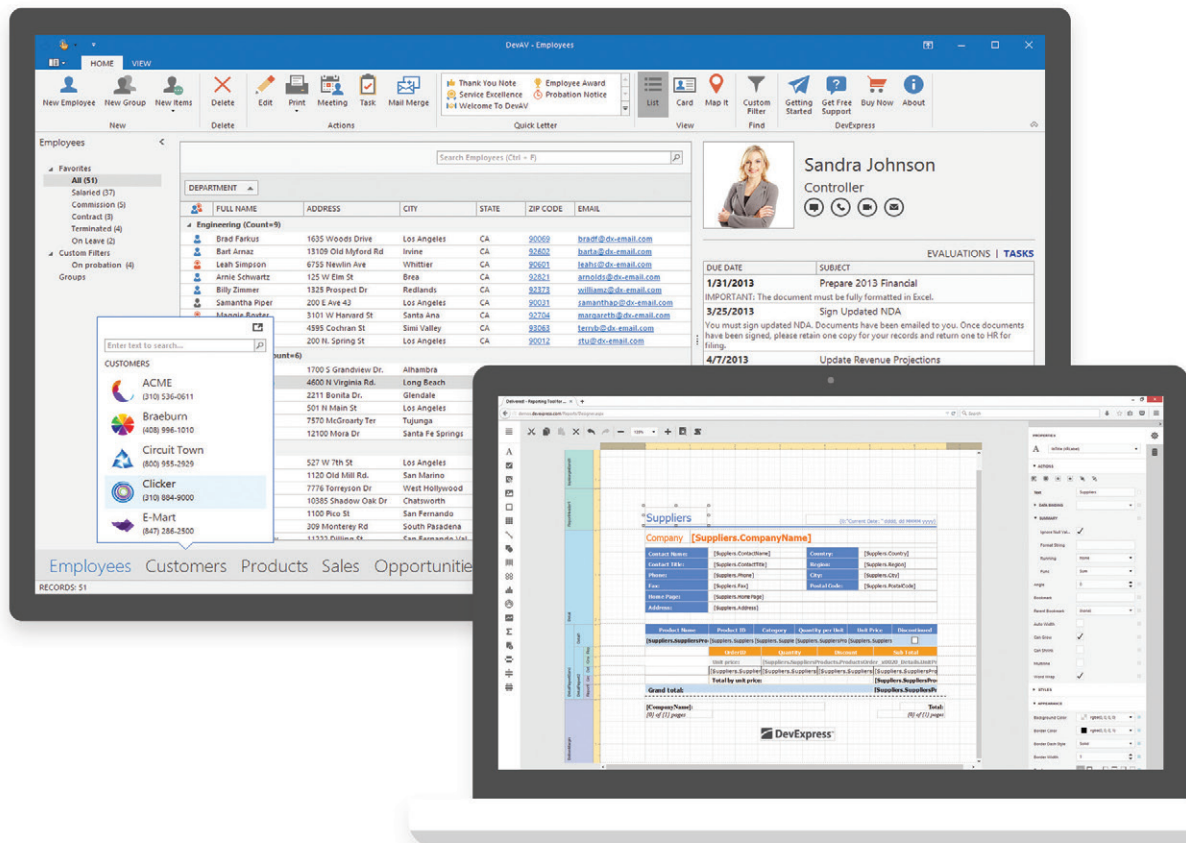
Free 30-Day Trial at
devexpress.com/trial





Your Next Great App Starts Here

From apps that replicate the look and feel of Microsoft Office® to high-powered data mining and decision support systems for your enterprise, DevExpress UI components for desktops, the web and mobile world will help you build your best, without limits or compromise.



Experience the DevExpress Difference
Download Your Free 30-Day Trial Today
devexpress.com/trial

All trademarks or registered trademarks are property of their respective owners.

msdn

magazine



Improvements
in ASP.NET Core 2.1.....20

ASP.NET Core:
What's New in ASP.NET Core 2.1
Steve Smith..... 20

Blockchain Fundamentals:
Diving into Transaction Hash Chains
Jonathan Waldman..... 28

HoloLens:
Mixed Reality and Fluent Design
Tim Kulp..... 40

COLUMNS

DATA POINTS

Deep Dive into EF Core
HasData Seeding
Julie Lerman, page 6

ARTIFICIALLY INTELLIGENT

Exploring the Custom
Vision Service
Frank La Vigne, page 14

CUTTING EDGE

Social-Style Notifications
with ASP.NET Core SignalR
Dino Esposito, page 50

TEST RUN

Introduction to Q-Learning
Using C#
James McCaffrey, page 56

DON'T GET ME STARTED

Sing in Me
David S. Platt, page 64

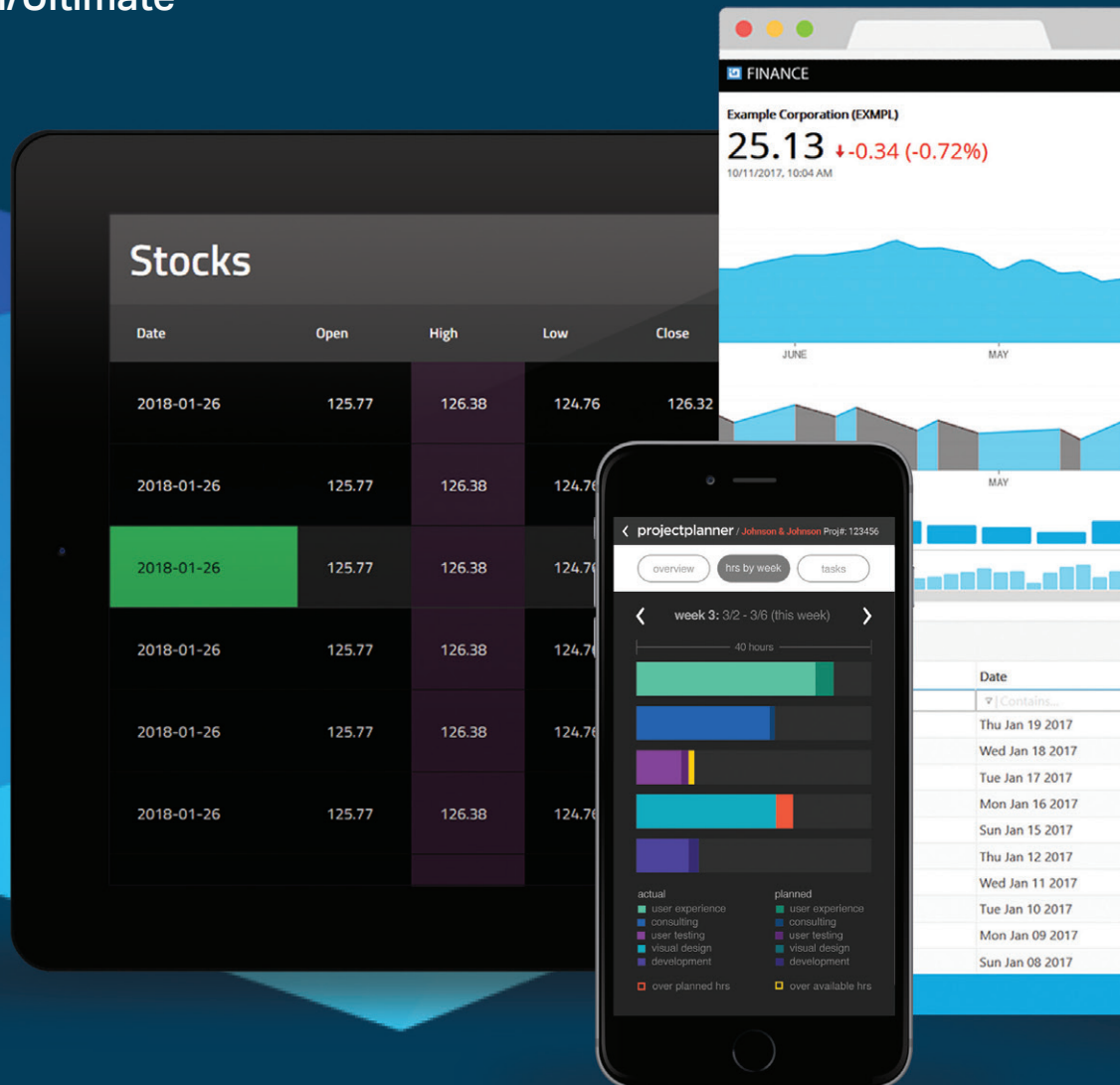


Faster Paths to Amazing Experiences

Infragistics Ultimate includes 100+ beautifully styled, high performance grids, charts & other UI controls, plus productivity tools for building web, desktop and mobile apps.

Angular | JavaScript / HTML5 | ASP.NET | Windows Forms | WPF | Xamarin

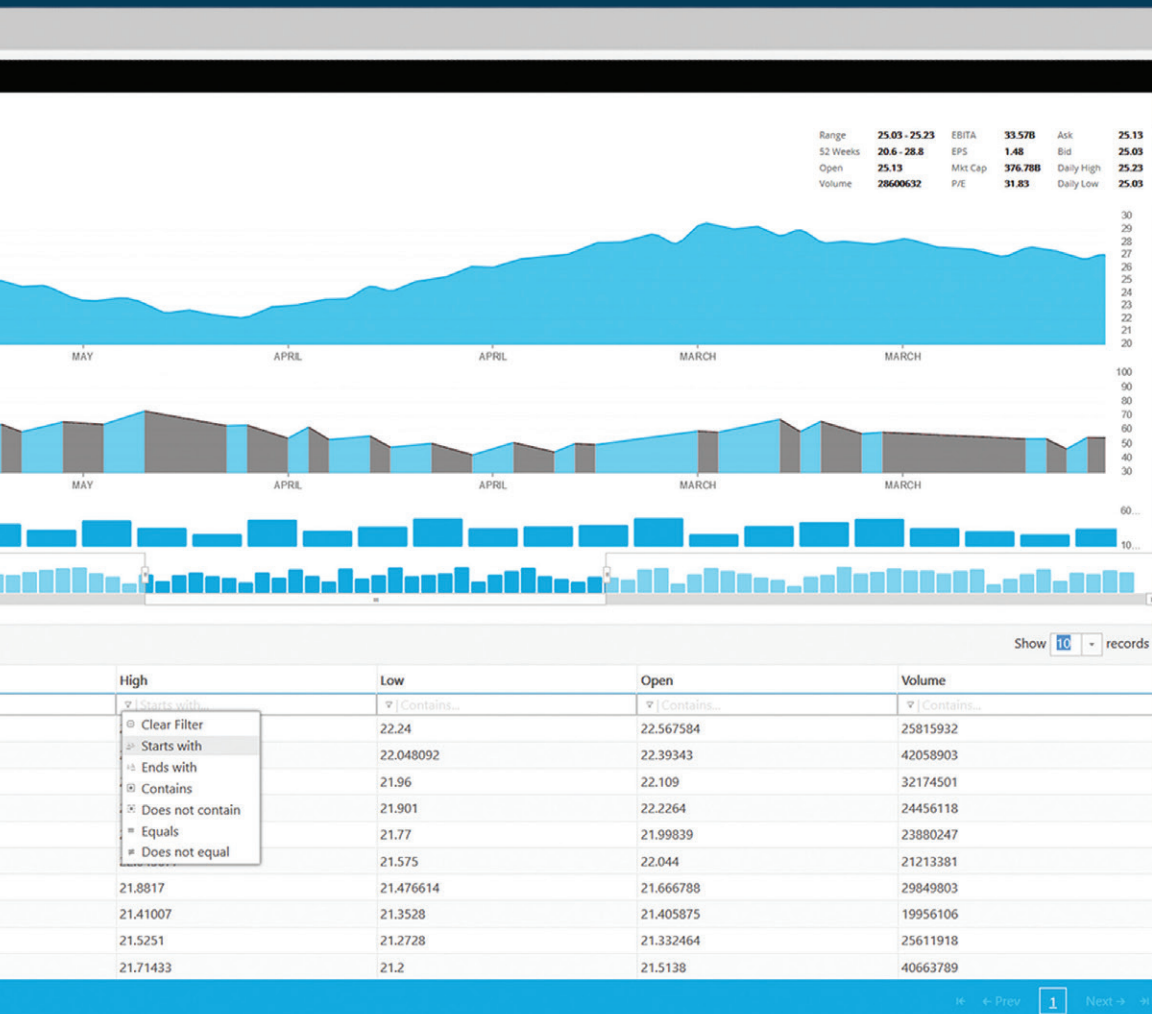
Get started today with a free trial:
[Infragistics.com/Ulimate](https://www.infragistics.com/Ulimate)



New Release

Infragistics Ultimate 18.1

- ✓ Fastest **grids & charts** on the market for the Angular developer
- ✓ The most complete **Microsoft Excel & Spreadsheet Solution** for .NET & JavaScript
- ✓ UI controls designed to meet the demands of the toughest **financial & capital market apps**



General Manager Jeff Sandquist

Director Dan Fernandez

Editorial Director Jennifer Mashkowski mmeditor@microsoft.com

Site Manager Kent Sharkey

Editorial Director, Enterprise Computing Group Scott Bekker

Editor in Chief Michael Desmond

Features Editor Sharon Terdeman

Group Managing Editor Wendy Hernandez

Senior Contributing Editor Dr. James McCaffrey

Contributing Editors Dino Esposito, Frank La Vigne, Julie Lerman, Mark Michaelis, Ted Neward, David S. Platt

Vice President, Art and Brand Design Scott Shultz

Art Director Joshua Gould



Chief Revenue Officer
Dan LaBianca

ART STAFF

Creative Director Jeffrey Langkau
Associate Creative Director Scott Rovin
Art Director Michele Singh
Art Director Chris Main
Senior Graphic Designer Alan Tao
Senior Web Designer Martin Peace

PRODUCTION STAFF

Print Production Manager Peter B. Weller
Print Production Coordinator Lee Alexander

ADVERTISING AND SALES

Chief Revenue Officer Dan LaBianca
Regional Sales Manager Christopher Kourtoglou
Advertising Sales Associate Tanya Egenolf

ONLINE/DIGITAL MEDIA

Vice President, Digital Strategy Becky Nagel
Senior Site Producer, News Kurt Mackie
Senior Site Producer Gladys Rama
Site Producer, News David Ramel
Director, Site Administration Shane Lee
Front-End Developer Anya Smolinski
Junior Front-End Developer Casey Rysavy
Office Manager & Site Assoc. James Bowling

LEAD SERVICES

Vice President, Lead Services Michele Imgrund
Senior Director, Audience Development & Data Procurement Annette Levee
Director, Audience Development & Lead Generation Marketing Irene Fincher
Director, Client Services & Webinar Production Tracy Cook
Director, Lead Generation Marketing Eric Yoshizuru
Senior Program Manager, Client Services & Webinar Production Chris Flack
Project Manager, Lead Generation Marketing Mahal Ramos

ENTERPRISE COMPUTING GROUP EVENTS

Vice President, Events Brent Sutton
Senior Director, Operations Sara Ross
Senior Director, Event Marketing Mallory Bastionell
Senior Manager, Events Danielle Potts
Senior Marketing Coordinator, Events Michelle Cheng



Chief Executive Officer
Rajeev Kapur

Chief Financial Officer
Craig Rucker

Chief Technology Officer
Erik A. Lindgren

Executive Vice President
Michael J. Valenti

Chairman of the Board
Jeffrey S. Klein

ID STATEMENT MSDN Magazine (ISSN 1528-4859) is published 13 times a year, monthly with a special issue in November by 1105 Media, Inc., 9201 Oakdale Avenue, Ste. 101, Chatsworth, CA 91311. Periodicals postage paid at Chatsworth, CA 91311-9998, and at additional mailing offices. Annual subscription rates payable in US funds are: U.S. \$35.00, International \$60.00. Annual digital subscription rates payable in U.S. funds are: U.S. \$25.00, International \$25.00. Single copies/back issues: U.S. \$10, all others \$12. Send orders with payment to: MSDN Magazine, P.O. Box 3167, Carol Stream, IL 60132, email MSDNmag@1105service.com or call (847) 763-9560. POSTMASTER: Send address changes to MSDN Magazine, P.O. Box 2166, Skokie, IL 60076. Canada Publications Mail Agreement No: 40612608. Return Undeliverable Canadian Addresses to Circulation Dept. or XPO Returns: P.O. Box 201, Richmond Hill, ON L4B 4R5, Canada.

Printed in the U.S.A. Reproductions in whole or part prohibited except by written permission. Mail requests to "Permissions Editor," c/o MSDN Magazine, 4 Venture, Suite 150, Irvine, CA 92618.

LEGAL DISCLAIMER The information in this magazine has not undergone any formal testing by 1105 Media, Inc. and is distributed without any warranty expressed or implied. Implementation or use of any information contained herein is the reader's sole responsibility. While the information has been reviewed for accuracy, there is no guarantee that the same or similar results may be achieved in all environments. Technical inaccuracies may result from printing errors and/or new developments in the industry.

CORPORATE ADDRESS 1105 Media, 9201 Oakdale Ave. Ste 101, Chatsworth, CA 91311 1105media.com

MEDIA KITS Direct your Media Kit requests to Chief Revenue Officer Dan LaBianca, 972-687-6702 (phone), 972-687-6799 (fax), dlabianca@1105media.com

REPRINTS For single article reprints (in minimum quantities of 250-500), e-prints, plaques and posters contact: PARS International
Phone: 212-221-9595
E-mail: 1105reprints@parsintl.com
Web: 1105Reprints.com

LIST RENTAL This publication's subscriber list is not available for rental. However, other lists from 1105 Media, Inc. can be rented.

For more information, please contact our list manager: Jane Long, Merit Direct
Phone: 913-685-1301;
E-mail: jloug@meritdirect.com;
Web: meritdirect.com/1105

Reaching the Staff

Staff may be reached via e-mail, telephone, fax, or mail. E-mail: To e-mail any member of the staff, please use the following form: FirstInitialLastname@1105media.com
Irvine Office (weekdays, 9:00 a.m. – 5:00 p.m. PT)
Telephone 949-265-1520; Fax 949-265-1528
4 Venture, Suite 150, Irvine, CA 92618
Corporate Office (weekdays, 8:30 a.m. – 5:30 p.m. PT)
Telephone 818-814-5200; Fax 818-734-1522
9201 Oakdale Avenue, Suite 101, Chatsworth, CA 91311
The opinions expressed within the articles and other contents herein do not necessarily express those of the publisher.



LEADTOOLS®

MULTI-PLATFORM BARCODE SDK



SDKs for
WEB



SDKs for
MOBILE



SDKs for
DESKTOP



SDKs for
SERVER



Use LEADTOOLS multi-platform Barcode SDK to develop applications that require reading and writing **1D** and **2D barcodes**. With comprehensive support of over 100 different barcode types such as UPC, EAN, Code 39, Code 128, ITF (2 of 5), CodaBar, GS1 DataBar, QR, Data Matrix, PDF417, and Aztec, LEADTOOLS is the right choice.



Get Started Today

DOWNLOAD OUR FREE EVALUATION

[LEADTOOLS.COM](https://leadtools.com)



Cover of the Rolling Stone

Want to see my picture on the cover
Wanna buy five copies for my mother
Wanna see my smilin' face
On the cover of the Rollin' Stone

—"Cover of the Rolling Stone," Dr. Hook & the Medicine Show

If you're old enough to remember the song "Cover of the Rolling Stone," you know it lampooned an aspiring rock band's dream to appear on the cover of the iconic music magazine *Rolling Stone*. That song, marked by the band's trademark humor, rose as high as No. 6 on the music charts. And in March 1973, the magazine famously took up the challenge. It put Dr. Hook & the Medicine Show on the cover—but in caricature, and with the cover line "What's- Their-Names Make the Cover."

Sometimes, life lampoons art imitating life.

So, what's it take to get on the
cover of *MSDN Magazine*? It
starts with a good, original idea.

Here at *MSDN Magazine* we've been putting talented developers on the cover since the days of DOS and the 640KB memory limit. Look through the articles in any issue of *MSDN Magazine*, and you'll find that much—even most—of the content is written by authors who work outside of Microsoft. This is not an accident. Our mission is to provide actionable, code-level guidance to working developers engaged with the Microsoft stack. And what better source for this technical wisdom than the developers in the field, who use the tools and frameworks, and brush up against sharp edges and lurking pain points every day?

We're here to give talented developers a platform to share their hard-won insights with the dev community. Do you have an innovative

approach or elegant solution to a common challenge? Have you found a way to leverage a new feature or tool that hasn't been widely explored? It could make for a compelling feature article.

"Don't tell them what. Show them how."

So, what's it take to get on the cover of *MSDN Magazine*? It starts with a good, original idea. The best concepts are both technically specific and broadly relevant. This can be tough (sometimes even impossible) to achieve, but proposals that manage this trick often do very well with our readers.

Avoid rote overviews or rehashes of published documentation or blog posts, and be sure your idea is technically rigorous. Our readers are veteran coders who've been around the block—they're seeking code-level insight. Finally, your idea must be robust enough stand up as a magazine feature, which typically runs three to five pages (about 2,400 to 3,400 words).

When it comes to structure, I advise authors to cast their article in terms of a realistic challenge/solution scenario if they can. This helps ground the exploration and makes it tangible for readers. As I always tell new authors to the magazine: "Don't tell them what. Show them how."

Keep in mind also that the best articles are *articles*, not white papers or documentation or technical blog posts. I urge authors to incorporate narrative elements like an engaging lead/intro, and a coherent "plot" that lays out the technical challenge and guides readers through its resolution.

Don't panic. We have skilled editors here who can help with all of that. But keeping these things in mind will help improve both your proposal and your finished article.

If you have an idea you'd like to propose, we invite you to submit your pitch at mmsubmit@microsoft.com. Just make sure the subject line of the e-mail reads "MSDN Article Query," so we're sure to see it. For detailed guidance on writing a successful proposal, also be sure to check out bit.ly/2L2fmDN.

It just may put you on the cover of *MSDN Magazine*.

Visit us at msdn.microsoft.com/magazine. Questions, comments or suggestions for *MSDN Magazine*? Send them to the editor: mmeditor@microsoft.com.

© 2018 Microsoft Corporation. All rights reserved.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, you are not permitted to reproduce, store, or introduce into a retrieval system *MSDN Magazine* or any part of *MSDN Magazine*. If you have purchased or have otherwise properly acquired a copy of *MSDN Magazine* in paper format, you are permitted to physically transfer this paper copy in unmodified form. Otherwise, you are not permitted to transmit copies of *MSDN Magazine* (or any part of *MSDN Magazine*) in any form or by any means without the express written permission of Microsoft Corporation.

A listing of Microsoft Corporation trademarks can be found at microsoft.com/library/toolbar/3.0/trademarks/en-us.mspx. Other trademarks or trade names mentioned herein are the property of their respective owners.

MSDN Magazine is published by 1105 Media, Inc. 1105 Media, Inc. is an independent company not affiliated with Microsoft Corporation. Microsoft Corporation is solely responsible for the editorial contents of this magazine. The recommendations and technical guidelines in *MSDN Magazine* are based on specific environments and configurations. These recommendations or guidelines may not apply to dissimilar configurations. Microsoft Corporation does not make any representation or warranty, express or implied, with respect to any code or other information herein and disclaims any liability whatsoever for any use of such code or other information. *MSDN Magazine*, MSDN and Microsoft logos are used by 1105 Media, Inc. under license from owner.

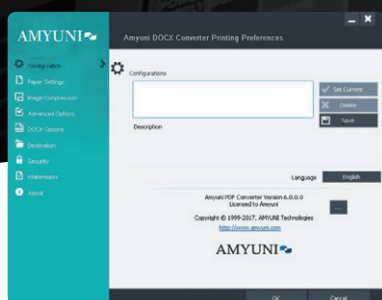


DOCXCONVERTER
For Windows

Free Demo at DOCXConverter.com

Amyuni DOCX Converter for Windows

Convert any document, including PDF documents, into DOCX format.
Enable editing of documents using Microsoft Word or other Office products.



A standalone desktop version, a server product for automated processing or an SDK for integration into third party applications.

Create

Create naturally editable DOCX documents with paragraph formatting and reflow of text

Convert

Convert images and graphics of multiple formats into DOCX shapes

OCR

Use OCR technology to convert non-editable text into real text

Extract

Extract headers and footers from source document and save them as DOCX headers and footers

Open

Open PDF documents with the integrated PDF viewer and quickly resave them to DOCX format

Configure

Configure the way the fonts are embedded into the DOCX file for optimal formatting

A virtual printer driver available for Windows 7 to Windows 10 and Windows Server 2008 to 2016

Powered by Amyuni Technologies:

Developers of the Amyuni PDF Converter and Amyuni PDF Creator products integrated into hundreds of applications and installed on millions of desktops and servers worldwide.

www.docxconverter.com

All trademarks are property of their respective owners. © Amyuni Technologies Inc. All rights reserved.



Deep Dive into EF Core HasData Seeding

The ability to seed data when migrations are run is a feature that disappeared in the transition from Entity Framework 6 (EF6) to Entity Framework Core (EF Core). With the latest version of EF Core, 2.1, seeding has made a comeback, yet in a very different form. In this article, you'll learn how the new seeding feature works, as well as scenarios where you may or may not want to use it. Overall, the new mechanism of seeding is a really handy feature for models that aren't overly complex and for seed data that remains mostly static once it's been created.

Basic Seeding Functionality

In EF6 (and earlier versions) you added logic into the `DbMigrationConfiguration.Seed` method to push data into the database any time migrations updated the database. For a reminder of what that looks like, check out the Microsoft ASP.NET documentation on seeding with EF6 at bit.ly/2ycTAlm.

In EF Core 2.1, the seeding workflow is quite different. There is now Fluent API logic to define the seed data in `OnModelCreating`. Then, when you create a migration, the seeding is transformed into migration commands to perform inserts, and is eventually transformed into SQL that that particular migration executes. Further migrations will know to insert more data, or even perform updates and deletes, depending on what changes you make in the `OnModelCreating` method.

You can use `HasData` to insert multiple rows at a time, though keep in mind that `HasData` is specific to a single entity.

If you happened to read the July 2018 Data Points (msdn.com/magazine/mt847184), you may recall the `Publications` model I used to demonstrate query types. I'll use that same model for these examples. In fact, I slid some data seeding into the July download sample! I'll start from a clean slate here, though.

The three classes in my model are `Magazine`, `Article` and `Author`. A magazine can have one or more articles and an article can have

one author. There's also a `PublicationsContext` that uses SQLite as its data provider and has some basic SQL logging set up.

Seeding Data for a Single Entity Type

Let's start by seeing what it looks like to provide seed data for a magazine—at its simplest.

The key to the new seeding feature is the `HasData` Fluent API method, which you can apply to an Entity in the `OnModelCreating` method.

Here's the structure of the `Magazine` type:

```
public class Magazine
{
    public int MagazineId { get; set; }
    public string Name { get; set; }
    public string Publisher { get; set; }
    public List<Article> Articles { get; set; }
}
```

It has a key property, `MagazineId`, two strings and a list of `Article` types. Now let's seed it with data for a single magazine:

```
protected override void OnModelCreating (ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Magazine> ().HasData
        (new Magazine { MagazineId = 1, Name = "MSDN Magazine" });
}
```

A couple things to pay attention to here: First, I'm explicitly setting the key property, `MagazineId`. Second, I'm not supplying the `Publisher` string.

Next, I'll add a migration, my first for this model. I happen to be using Visual Studio Code for this project, which is a .NET Core app, so I'm using the CLI migrations command, "dotnet ef migrations add init." The resulting migration file contains all of the usual `CreateTable` and other relevant logic, followed by code to insert the new data, specifying the table name, columns and values:

```
migrationBuilder.InsertData(
    table: "Magazines",
    columns: new[] { "MagazineId", "Name", "Publisher" },
    values: new object[] { 1, "MSDN Magazine", null });
```

Inserting the primary key value stands out to me here—especially after I've checked how the `MagazineId` column was defined further up in the migration file. It's a column that should auto-increment, so you may not expect that value to be explicitly inserted:

```
MagazineId = table.Column<int>(nullable: false)
    .Annotation("Sqlite:Autoincrement", true)
```

Let's continue to see how this works out. Using the migrations script command, "dotnet ef migrations script," to show what will be sent to the database, I can see that the primary key value will still be inserted into the key column:

```
INSERT INTO "Magazines" ("MagazineId", "Name", "Publisher")
VALUES (1, 'MSDN Magazine', NULL);
```

That's because I'm targeting SQLite. SQLite will insert a key value if it's provided, overriding the auto-increment. But what about with

Code download available at msdn.com/magazine/0818magcode.



Telerik + Kendo UI Modern UI Made Easy

Q&A with Stefan Stefanov, Senior Product Manager

Q What sets Progress Telerik apart from the other component vendors?

A It's simple—modern UI at industry speed and supporting you as a developer along the way. Unlike other vendors we support both new UI interfaces such as chatbots or AR/VR, in addition to legacy and mainstream platforms from .NET and JavaScript spaces. Telerik DevCraft contains more than 1,100 Telerik .NET and Kendo UI JavaScript components, reporting, productivity and testing tools. We have award-winning support, free product training, and hundreds of demos and sample apps to get you started easily and quickly.

Q Tell us more about those Modern UI tools you mention?

A We spend time on the bleeding edge so you know that when you get there, we will have perfected tools you need for your maximum productivity. Our latest innovations include Conversational UI—the industry's first set of chatbot framework-agnostic user interface controls that enable .NET and JavaScript developers to create chatbot experiences in their web, desktop, and mobile applications—and AR/VR controls, which are in pre-alpha stage.

Q What technologies and platforms do you support?

A Our Telerik .NET tools cover ASP.NET AJAX, MVC and Core, Xamarin, WPF, WinForms, UWP and Silverlight. Our Kendo UI JavaScript tools support Angular, jQuery, React, Vue, JSP and PHP.

Q Telerik got its start with ASP.NET AJAX. What is your view of the Web and Web Forms?

A Our view of the web is bigger than ever. We still support Web Forms through our Telerik UI for ASP.NET AJAX tools. We also



provide tooling for some of the more popular JavaScript libraries like Angular, jQuery, React and Vue; and frameworks like ASP.NET MVC, ASP.NET Core. We are constantly evaluating what our customers are leaning towards to ensure we have the tooling ready when they make the leap. We do this without abandoning the tools our customers are currently using.

Q Microsoft has been active with OSS. Where does Progress stand on Open Source?

A We are a big believer in Open Source and have released the Telerik UI for UWP suite to the community as open source and contributed all the code to the .NET Foundation. Progress is also the creator of NativeScript—the open source framework for building truly native mobile apps with Angular, Vue.js, TypeScript or JavaScript. There is a vibrant community of developers contributing to both projects.

For more information, please visit →

www.telerik.com/msdn

a SQL Server database, which definitely won't do that on the fly?

I switched the context to use the SQL Server provider to investigate and saw that the SQL generated by the SQL Server provider includes logic to temporarily set `IDENTITY_INSERT` ON. That way, the supplied value will be inserted into the primary key column. Mystery solved!

You can use `HasData` to insert multiple rows at a time, though keep in mind that `HasData` is specific to a single entity. You can't combine inserts to multiple tables with `HasData`. Here, I'm inserting two magazines at once:

```
modelBuilder.Entity<Magazine>()
    .HasData(new Magazine(MagazineId=2, Name="New Yorker"),
              new Magazine(MagazineId=3, Name="Scientific American"))
    );
```

What About Non-Nullable Properties?

Remember that I've been skipping the `Publisher` string property in the `HasData` methods, and the migration inserts null in its place. However, if I tell the model builder that `Publisher` is a required property, in other words, that the database column is non-nullable, `HasData` will enforce that.

Here's the `OnModelCreating` code I've added to require `Publisher`:

```
modelBuilder.Entity<Magazine>().Property(m=>m.Publisher).IsRequired();
```

Now, when I try to add a migration to account for these new changes (the `IsRequired` method and seeding two more magazines), the migrations add command fails, with a very clear error message:

```
"The seed entity for entity type 'Magazine' cannot be added because there
was no value provided for the required property 'Publisher'."
```

This happened because the two new magazines I'm adding don't have a `Publisher` value.

The same would happen if you tried to skip the `MagazineId` because it's an integer, even though you know that the database will provide the value. EF Core also knows that the database will generate this value, but you're still required to provide it in `HasData`.

The need to supply required values leads to another interesting limitation of the `HasData` feature, which is that there's a possibility it will conflict with a constructor. Imagine I have a constructor for `Magazine` that takes the magazine's name and publisher's name:

```
public Magazine(string name, string publisher)
{
    Name=name;
    Publisher=publisher;
}
```

As the database will create the key value (`MagazineId`), there's no reason I'd have `MagazineId` as a parameter of such a constructor.

Thanks to another new feature of EF Core 2.1, I no longer have to add a parameterless constructor to the class in order for queries to materialize magazine objects. That means the constructor is the only option for me to use in my `HasData` method:

```
modelBuilder.Entity<Magazine>()
    .HasData(new Magazine("MSDN Magazine", "1105 Media"));
```

But, again, this will fail because I'm not supplying a value for the non-nullable `MagazineId` property. There's a way around this, however, which takes advantage of the EF Core shadows property feature—using anonymous types instead of explicit types.

HasData with Anonymous Types

The ability to seed with anonymous types instead of explicit types solves a lot of potential roadblocks with `HasData`.

The first is the one I just explained, where I created a constructor for the `Magazine` class and there's no way to set the non-nullable `MagazineId` when seeding with `HasData`. Instead of instantiating a `Magazine`, you can instantiate an anonymous type and supply the `MagazineId`, without worrying about whether the property is public or private, or even exists! That's what I'm doing in the following method call:

```
modelBuilder.Entity<Magazine>()
    .HasData(new {MagazineId=1, Name="MSDN Mag", Publisher="1105 Media"});
```

The migration code will correctly insert that data into the `magazines` table, and running the migrations update database command works as expected:

```
migrationBuilder.InsertData(
    table: "Magazines",
    columns: new[] { "MagazineId", "Name", "Publisher" },
    values: new object[] { 1, "MSDN Mag", "1105 Media" });
```

You'll see a few more roadblocks that the anonymous type solves further on.

What About Private Setters?

The limitation posed by the required primary key stands out because `Magazine` uses an integer as a key property. I've written many solutions, however, that use `Guids` for keys and my domain logic ensures that a `Guid` value is created when I instantiate an entity. With this setup, I can protect any properties by using private setters, yet still get the key property populated without exposing it. But there's a problem for `HasData`. First, let's see the effect and then explore (and solve) the problem.

As an example, I've transformed `Magazine` in **Figure 1** so that `MagazineId` is a `Guid`, the setters are private and the only way (so far) to set their values is through the one and only constructor.

Now I'm assured that when I create a new `Magazine` object a `MagazineId` value will be created, as well:

```
modelBuilder.Entity<Magazine>().HasData(new Magazine("MSDN Mag", "1105 Media"));
```

The migration generates the following `InsertData` method for `Magazine`, using the `Guid` created in the constructor:

```
migrationBuilder.InsertData(
    table: "Magazines",
    columns: new[] { "MagazineId", "Name", "Publisher" },
    values: new object[] { new Guid("8912aa35-1433-48fe-ae72-de2aaa38e37e"),
                          "MSDN Mag", "1105 Media" });
```

However, this can cause a problem for the migration's ability to detect changes to the model. That `Guid` was auto-generated when I created the new migration. The next time I create a migration a different `Guid` will be generated and EF Core will see this as a change to the data, delete the row created from the first

Figure 1 The Magazine Type with a Guid Key, Private Setters and a Parameter Constructor

```
public class Magazine
{
    public Magazine(string name, string publisher)
    {
        Name=name;
        Publisher=publisher;
        MagazineId=Guid.NewGuid();
    }
    public Guid MagazineId { get; private set; }
    public string Name { get; private set; }
    public string Publisher { get; private set; }
    public List<Article> Articles { get; set; }
}
```

Powerful In-Memory Computing Platform for .NET Applications

Go beyond distributed caching: now .NET applications can easily track and analyze fast-changing data, including event streams, to extract new insights and provide feedback in real time.

Beyond Distributed Caching

As the need to quickly track fast-changing data has steadily grown, .NET applications have deployed in-memory data grids (IMDGs) as distributed caches to ensure fast data access and meet scalability requirements. However, IMDGs now have exciting new in-memory computing capabilities which can take application performance to the next level while simplifying design and eliminating bottlenecks.

Founded by Microsoft alumni and focused on the needs of .NET developers since 2005, ScaleOut Software brings the power of parallel supercomputing to .NET applications with its comprehensive suite of distributed software, including a powerful IMDG, stateful stream-processing, and data-parallel analytics.

Battle-Tested In-Memory Data Grid

The foundation of ScaleOut's software suite is its IMDG, ScaleOut StateServer®, which has been proven in mission-critical applications for more than a decade. Designed to combine speed and ease of use, this IMDG offers a rich feature set built on a highly scalable, peer-to-peer architecture. Key distinguishing capabilities include sequentially coherent data access under all conditions, transparent load-balancing, and patented, quorum-based, high availability.

Stateful Stream-Processing

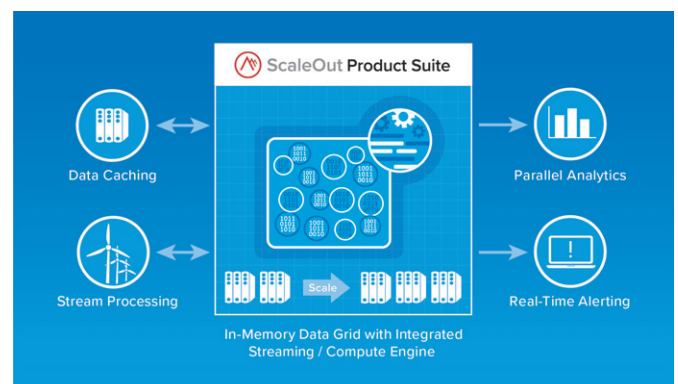
Made possible by combining IMDGs with in-memory computing, the next generation of stream-processing has arrived, and it's enabling deeper introspection on streaming data with faster responses than ever before. As an industry-leading implementation of the popular "digital twin" model, ScaleOut StreamServer™ combines an IMDG with an integrated streaming/compute engine, ReactiveX APIs, and Kafka integration to quickly and easily correlate incoming

events with context information about their data sources for stateful stream-processing. This powerful new architecture eliminates network bottlenecks and maximizes scalability. Applications include IoT, financial services, ecommerce, logistics, and many others.

Data-Parallel Analytics

By design, IMDGs host fast-changing data. With its integrated in-memory compute engine, ScaleOut's IMDG lets users analyze this data fast using data-parallel APIs such as MapReduce and Distributed ForEach, an extension of .NET's Parallel.ForEach API. Now applications can identify important patterns in their data in real time and with maximum performance, eliminating the need for offline, batch processing. By going beyond distributed caching, ScaleOut's integrated IMDG and compute engine creates new opportunities for applications to extract value from their real-time data and react fast—before the moment is lost.

To learn more about ScaleOut's product suite and download a free 30-day trial, please visit www.scaleoutsoftware.com/products



migration and insert a new row using the new Guid. Therefore, you should use explicit Guids when seeding with HasData, never generated ones. Also, you'll need to use the anonymous type again, rather than the constructor, because MagazineId doesn't have a public setter:

```
var mag1=new {MagazineId= new Guid("0483b59c-f7f8-4b21-b1df-5149fb57984e"),
    Name="MSDN Mag", Publisher="1105 Media"};
modelBuilder.Entity<Magazine>().HasData(mag1);
```

Keep in mind that explicitly creating Guids in advance could get cumbersome with many rows.

Seeding Related Data

Using HasData to seed related data is very different from inserting related data with DbSets. I stated earlier that HasData is specific to a single entity. That means you can't build graphs as parameters of HasData. You can only provide values for properties of that Entity.

Therefore, if you want to add a Magazine and an Article, these tasks need to be performed with separate HasData methods—one on Entity<Magazine> and one on Entity<Article>. Take a look at the schema of the Article class:

```
public class Article
{
    public int ArticleId { get; set; }
    public string Title { get; set; }
    public int MagazineId { get; set; }
    public DateTime PublishDate { get; set; }
    public int? AuthorId { get; set; }
}
```

Notice that the MagazineId foreign key is an int and, by default, that's non-nullable. No article can exist without a Magazine identified. However, the AuthorId is a nullable int, therefore it's possible to have an article that hasn't yet had an author assigned. This means that when seeding an Article, in addition to the required ArticleId value, you must supply the MagazineId. But you're not required to supply the AuthorId. Here's code to add an article where I've supplied the key value (1), the value of an existing magazine's ID (1) and a Title—I didn't provide an AuthorId or a date:

```
modelBuilder.Entity<Article>().HasData(
    new Article { ArticleId = 1, MagazineId = 1,
        Title = "EF Core 2.1 Query Types" });
```

The resulting migration code is as follows:

```
migrationBuilder.InsertData(
    table: "Articles",
    columns: new[] { "ArticleId", "AuthorId", "MagazineId", "PublishDate", "Title" },
    values: new object[] { 1, null, 1, new DateTime(1, 1, 1, 0, 0, 0, 0,
        DateTimeKind.Unspecified), "EF Core 2.1 Query Types" });
```

The migration is adding null for AuthorId, which is fine. I didn't supply a PublishDate, so it's defaulting to the minimal .NET date value (01/01/0001). If I were to add a MagazineId or an AuthorId that doesn't yet exist, it won't be caught until the SQL is run against the database, triggering a referential integrity error.

If you've followed my work for a while, you may know that I'm a big fan of using foreign key properties for related data, rather than navigation properties. But there are scenarios where you may prefer not to have the foreign key property in your dependent type. EF Core can handle that thanks to shadow properties. And once again, anonymous types come to the rescue with HasData to seed the related data that requires you to supply the value of the foreign key column.

Seeding Owned Entities

Owned entities, also known as owned types, are the way EF Core lets you map non-entity types, replacing the complex type feature of Entity Framework. I wrote about this new support in the April 2018 Data Points column (msdn.com/magazine/mt846463). Because an owned type is specific to the entity that owns it, you'll need to do the data seeding as part of the definition of the type as a property of an entity. You can't just populate it from modelBuilder the way you do for an entity.

To demonstrate, I'll introduce a new type in my model, Publisher:

```
public class Publisher
{
    public string Name { get; set; }
    public int YearFounded { get; set; }
}
```

Notice it has no key property. I'll use Publisher as a property of Magazine in place of the Publisher string and, at the same time, revert to a simpler Magazine class:

```
public class Magazine
{
    public int MagazineId { get; set; }
    public string Name { get; set; }
    public Publisher Publisher { get; set; }
    public List<Article> Articles { get; set; }
}
```

Two important points to remember are that you can only provide properties for one entity type with HasData and that the Model Builder treats an owned type as a separate entity. In this case, that means you can't populate a magazine and its publisher in a single Entity<Magazine>.HasData method. Instead, you have to identify the owned property (even if you've configured it elsewhere) and append HasData to it.

You may prefer not to have the foreign key property in your dependent type. EF Core can handle that thanks to shadow properties.

I'll first provide some Magazine data:

```
modelBuilder.Entity<Magazine> ()
    .HasData (new Magazine { MagazineId = 1, Name = "MSDN Magazine" });
```

Seeding the owned type is a little tricky only because it may not be something you can intuit. Because the model builder will treat Publisher as a related object in order to persist it properly, it needs to know the value of the MagazineId that owns it. As there's no MagazineId property in Publisher—EF Core uses its shadow property feature to infer a MagazineId property. In order to set that property, you'll need to instantiate an anonymous type rather than a Publisher. If you tried to instantiate a Publisher, it wouldn't accept the MagazineId property in the initializer:

```
modelBuilder.Entity<Magazine> ()
    .OwnsOne (m => m.Publisher)
    .HasData (new { Name = "1105 Media", YearFounded = 2006, MagazineId=1 });
```




From Desktops to Web and Mobile Your Next Great App Starts Here

Experience the DevExpress difference and see why your peers consistently vote our products #1. With our Universal Subscription, you will build your best, see complex software with greater clarity, increase your productivity and create stunning applications for Windows, Web and your Mobile world.



DevExpress Universal ships with 500+ UI controls.
It also includes our royalty-free reporting and dashboard platform.

WIN ASP MVC WPF UWP JS

Download your free 30-day trial today.
devexpress.com/try

All trademarks or registered trademarks are property of their respective owners.

When I create a migration to take this pairing into account, the resulting `InsertData` method knows to insert all of the values—the properties of `Magazine` and its owned type, `Publisher`—into the `Magazine` table:

```
migrationBuilder.InsertData(
    table: "Magazines",
    columns: new[] { "MagazineId", "Name", "Publisher_Name", "Publisher_YearFounded" },
    values: new object[] { 1, "MSDN Magazine", "1105 Media", 2006 });
```

This works out easily enough when my classes are simple, although you may reach some limitations with more complicated classes.

No Migrations? `EnsureCreated` Does the Job

Finally, we've reached the point where you get to see the dual nature of the new seeding mechanism. When you're using database providers with migrations commands, the migrations will contain the logic to insert, update or delete seed data in the database. But at run time, there's only one way to trigger `HasData` to be read and acted upon, and that's in response to the `DbContext.Database.EnsureCreated` method. Keep in mind that `EnsureCreated` won't run migrations if the database already exists. The provider that really benefits from this is the `InMemory` provider. You can explicitly create and seed `InMemory` databases in your tests by calling `EnsureCreated`. Unlike the `Migrate` command, which runs migrations—and will execute and seed methods in those migrations—`EnsureCreated` creates a database using the model described by the context class. And whatever provider you're using, that will also cause `HasData` methods to insert data at the same time.

To demonstrate, I've modified the `PublicationsContext` by adding a new constructor to allow for injecting a provider by adding an explicit public parameterless constructor to allow for passing in pre-configured options:

```
public PublicationsContext (DbContextOptions<PublicationsContext> options) :
    base (options) { }
public PublicationsContext () { }
```

And I've added logic to skip the `UseSqlite` method in `OnConfiguring` if the options have already been configured:

```
protected override void OnConfiguring (DbContextOptionsBuilder
optionsBuilder)
{
    if (!optionsBuilder.IsConfigured)
    {
        optionsBuilder.UseSqlite (@".\Filename=Data\PubsTracker.db");
    }
    optionsBuilder.UseLoggerFactory (MyConsoleLoggerFactory);
}
```

Note that I moved the `UseLoggerFactory` command to run after the `IsConfigured` check. If it comes before the check, `IsConfigured` returns true. I started out that way and it took me a while to figure out what was wrong.

My automated test sets up the options to use the `InMemory` provider. Next, critical to the seeding, it calls `EnsureCreated` and then tests to see if there is, indeed, some data already available:

```
public void CanRetrieveDataCreatedBySeeding ()
{
    var options = new DbContextOptionsBuilder<PublicationsContext> ()
        .UseInMemoryDatabase ("RetrieveSeedData").Options;
    using (var context = new PublicationsContext (options))
    {
        context.Database.EnsureCreated();
        var storedMag = context.Magazines.FirstOrDefault ();
        Assert.Equal ("MSDN Magazine", storedMag.Name);
    }
}
```

The test passes because `EnsureCreated` forced the `HasData` methods to push the seed data into the `InMemory` database.

A Variety of Use Cases, but Not All of Them

Even though you've seen some of the limitations of using `HasData` in a few more-complex scenarios, it's definitely a nice improvement over the workflow that existed in earlier versions of EF. I really appreciate that I now have more control over the data flow by tying the insert, update and delete statements to individual migrations, rather than having to worry about upserts on every migration. The syntax is much cleaner, as well. Most important is the dual nature of this feature that not only allows you to get the seed data into your development (or even production) database, but also means that by calling `EnsureCreated`, you can seed the `InMemory` data to provide a consistent base of seed data that will be relevant for each test.

When you're using database providers with migrations commands, the migrations will contain the logic to insert, update or delete seed data in the database.

But `HasData` isn't a silver bullet. Keep in mind that this feature is best for seed data that will remain static once it's been inserted into the database. Also, watch out for `HasData` migrations that could override data you've seeded outside of migrations. As I explained earlier with the `Guids`, `HasData` doesn't work well with computed data. Andriy Svyryd from the EF team adds, "For seeding testing data that's not expected to be maintained between migrations or to have more complex logic, like computing seed values from the current state of the database, it's still possible and encouraged to just create a new instance of the context and add the data using `SaveChanges`." As another alternative, I've heard from readers that my method for seeding with JSON data is still working nicely, even with EF Core 2.1. I wrote about this in a blog post at bit.ly/2MvTyhM.

If you want to stay informed on how `HasData` will evolve, or even on issues that users are discovering, keep an eye on the GitHub repository at bit.ly/2l8VrEy and just filter on `HasData`. ■

JULIE LERMAN is a Microsoft Regional Director, Microsoft MVP, software team coach and consultant who lives in the hills of Vermont. You can find her presenting on data access and other topics at user groups and conferences around the world. She blogs at thedatafarm.com/blog and is the author of "Programming Entity Framework," as well as a *Code First* and a *DbContext* edition, all from O'Reilly Media. Follow her on Twitter: [@julielerman](https://twitter.com/julielerman) and see her *Pluralsight* courses at julieme/PS-Videos.

THANKS to the following Microsoft technical expert for reviewing this article:
Andriy Svyryd

File Format APIs

Open, Create, Convert, Print and Save files from your applications!

Try risk free – 30 day trial



Download a Free Trial at



<https://downloads.aspose.com>



Aspose.Words

Create, edit, convert or print Word documents (DOC, DOCX, RTF etc.) in your .NET, Java and Android applications.



Aspose.Cells

Develop high performance .NET, Java and Android applications to Create, Edit or Convert Excel worksheets (XLS, XLSX, ODS etc).



Aspose.Pdf

Manipulate PDF file formats (PDF, PDF/A, XPS etc.) using our native APIs for .NET, Java and Android platforms.



Aspose.Slides

Create, edit or convert PowerPoint presentations (PPT, PPTX, ODP etc.) in your .NET, Java and Android applications.



Aspose.Email

Create, Edit or Convert Outlook Email file formats (MSG, PST, EML etc.) and popular network protocols.



Aspose.BarCode

Generate or recognize barcodes (Code128, PDF417, Postnet etc.) using our native APIs for .NET and Java.



Aspose.Imaging

Deliver efficient applications to Create, Draw, Manipulate or Convert image file formats.



Aspose.Tasks

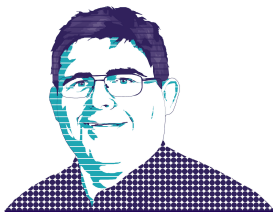
Develop high performance apps to Create, Edit or Convert Microsoft Project® document formats.

► Aspose.Diagram ► Aspose.Note ► Aspose.3D ► Aspose.CAD ► Aspose.HTML ► Aspose.GIS

Americas: +1 903 306 1676

EMEA: +44 141 628 8900
sales@asposeptyltd.com

Oceania: +61 2 8006 6987



Exploring the Custom Vision Service

One of the more fascinating cognitive services available in Microsoft Azure is the Custom Vision Service, which allows users to easily create custom computer vision models. Traditionally, training an image classifier required thousands, if not tens of thousands, of images per class to generate a model accurate enough for practical use. Additionally, you would also need a strong grasp on the mechanics of neural networks. The Custom Vision Service provides an easy-to-use Web UI that masks this underlying complexity.

Better still, the Custom Vision Service can create fairly accurate custom image classifiers with as little as 15 or 20 images per class, although the documentation recommends a minimum of 50 images per class. Generally, the quality of the image classifier will improve with more training images. The Custom Vision Service also exposes trained models via a REST API to simplify the deployment of these models.

At the Build 2018 conference, one of the keynote demos involved an unmanned aerial vehicle, or drone, flying over a series of pipes to detect defects and maintenance issues. In the demo, a drone streamed video from its camera back to a laptop. The laptop evaluated the images against a trained custom vision model, which alerted the drone operator of any defects seen in the pipes. For this article, I'll mimic a similar artificial intelligence (AI)-augmented maintenance model for detecting issues with train tracks. To do so, I'll explore the Custom Vision Service and demonstrate how easy it is to add custom image classification to any app or Web site.

Sourcing Images to Train On

Creating a computer vision model with the Custom Vision Service involves three simple steps:

1. Upload a set of images and label them.
2. Train and generate the model by clicking on the Train button.
3. Use the Web interface or REST APIs to evaluate the model's performance.

It's actually that easy to get started. But care must be taken as you begin sourcing images for your model on which to train.

Recently for a customer demo, I collected a series of images of train tracks via the Internet. These images included normal train tracks and those with severe defects. With as little as 30 images, I was able to generate a custom vision model to detect track defects with around 80 percent accuracy as part of my presentation. As impressive as that may be, I was quick to point out to the customer that any production-ready image classification model would require images of train tracks in a variety of diverse lighting

conditions, angles and so on. This diversity of classification data reduces the likelihood of the algorithm isolating incorrect features of the training images, which would wreak havoc with the results.

There's a classic, and possibly apocryphal, cautionary tale about training neural networks. The legend explains what researchers in the 1980s faced when the U.S. Army decided to train a neural network to detect tanks. According to the story, the neural network performed very well against the test data as part of the experiment. However, when the neural network was given new images to classify, it performed terribly.

The researchers were dumbfounded until they realized that all the images containing tanks had been taken on overcast days, while all the images lacking tanks were taken on sunny days. The neural network separated the two classes of photos and chose to

Figure 1 The Create New Project Dialog



The Current State of Data Quality – What Can You Do About It?

Q&A with Bud Walker, Vice President of Enterprise Sales & Strategy

Q What is the current state of data quality?

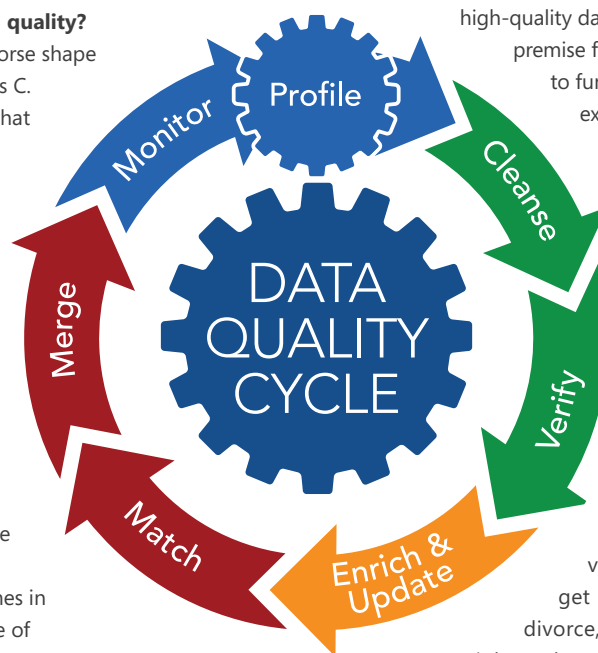
A Shockingly terrible. Data is in far worse shape than we realize. The data doc, Thomas C. Redman, recently released an article that stated only 3% of companies' data actually lives up to company standards. And unfortunately, everyone's experienced the trouble of bad data—poor business decisions, wasted time, increased costs and dissatisfied customers.

Q How do most companies approach data quality?

A Well, most companies aren't doing it right. The approach is reactive rather than getting in front of the problem. A large amount of data comes in flawed—nearly half actually—because of typographical errors or non-conforming data. And despite the headache and double work of fixing errors after they enter your database, companies still aren't looking into proactive solutions. But real-time verification solutions exist. Melissa's solutions catch discrepancies before entering your system—they validate international addresses as you type, make sure phone numbers are live, and ping emails, for example.

Q What opportunities come from having quality data at your fingertips?

A Endless. Quality data allows you to satisfy your customers, create marketplace advantage, cut costs, build stronger companies and empower people. We see forward-thinking, successful organizations and individuals racing towards Big Data analytics, AI [artificial intelligence], machine learning, machine reasoning. But, these new technologies mean nothing without clean, useful,



high-quality data at the core. Good data is the premise for these leading-edge technologies to function properly and live up to expectations. Data drives and empowers everything. It is the key to operating a successful business.

Q How does Melissa help companies achieve data quality to become more successful?

A Melissa has three decades of experience helping companies achieve data quality by providing a full-spectrum of solutions.

We start with contact data validation—the hardest aspect to get right as customers move, marry, divorce, or frequently change their email, job or phone number. So, we call in the Big 4: name, email, phone and address verification. Then we append and update incorrect contact info, standardize U.S. and international data sets, consolidate and remove duplicate records for a single view of the customer, and enrich with demographics, lifestyle, mortgage and property data, business firmographics and location intelligence, to bring deeper insight. The result is clean, accurate, reliable data for improved analytics, efficient operations and stronger customer relationships. And we make it easy—we offer off-the-shelf tools powered by multisource reference data so you don't have to reinvent the wheel. For those who love building or using a hybrid approach, we also offer every kind of integration option you can imagine. We invite you to play around in our sandbox if you will, the Melissa Developer Portal, where you can test out all of our tools, at the coding level!

For more information, please visit →

www.Melissa.com/Developer

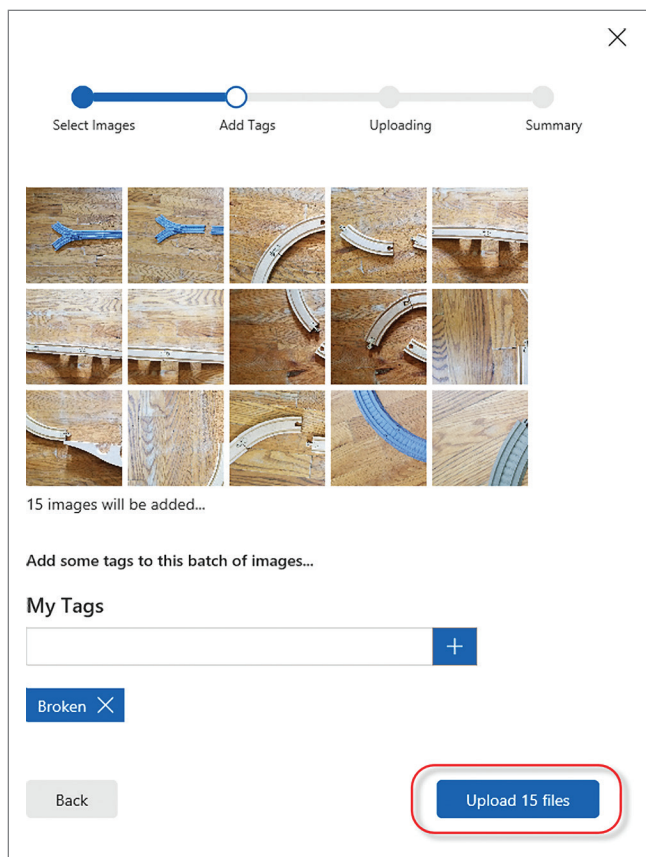


Figure 2 Tagged Broken Track Images

distinguish the two sets based on the color of the sky rather than the presence of a tank. Obviously, this is not what the Army needed. For a full exploration of the tank detection story, you can check out Neil Fraser's original write up at bit.ly/2llxudb, but this part of his article is worth quoting here:

"It is a perfect illustration of the biggest problem behind neural networks. Any automatically trained net with more than a few dozen neurons is virtually impossible to analyze and understand. One can't tell if a net has memorized inputs, or is 'cheating' in some other way."

Although the article was last updated in 2003, this quote still holds true today. Work is being done to better understand the inner workings of complex neural networks, but this effort is ongoing and is barely beyond its nascent phase. The best practice currently is to source data that contains a varied set of training data. This helps the algorithm distinguish images based on the correct variables.

To avoid image copyright issues, I've decided to leverage my children's large toy train track collection for this article. This also allows me to create training and testing imagery quickly. I've taken 34 images and split them into three folders named Broken, Normal and Test. Images with "broken" train tracks are in the Broken folder, and images with contiguous tracks are in the Normal folder. I've randomly chosen a series of images to test the model on and placed them in the Test folder. With my source images selected and labeled, it's time to create a model.

Creating the Model

In a browser, I go to customvision.ai and click on the Sign In button to sign in with my Microsoft account. Once logged in, I click on New Project to create a new custom vision project.

In the dialog that follows, I enter "Toy Trains" for the project name and a brief description of the project. I leave the Resource Group dropdown list to the default setting of Limited trial, and make sure the radio buttons for Project Types and Domains are set to Classification and General, respectively. Next, I click the Create project button to create a new custom vision project. Figure 1 shows these settings.

Uploading the Images

Once the project is created, the Web site prompts you to upload images. I click Add Images and in the following dialog click Browse local files to find images to upload. First, I upload the images in the Broken folder by selecting them all and clicking Open in the browsing dialog. In the text box below the images, I enter the term Broken and click the blue plus sign to tag the images with the Broken label. I then click the Upload button to upload the 15 images. This step is shown in Figure 2. Next, I click Done to finish this step.

Now that the images of broken train tracks are uploaded, it's time to upload the images in the Normal folder. To do this, I click

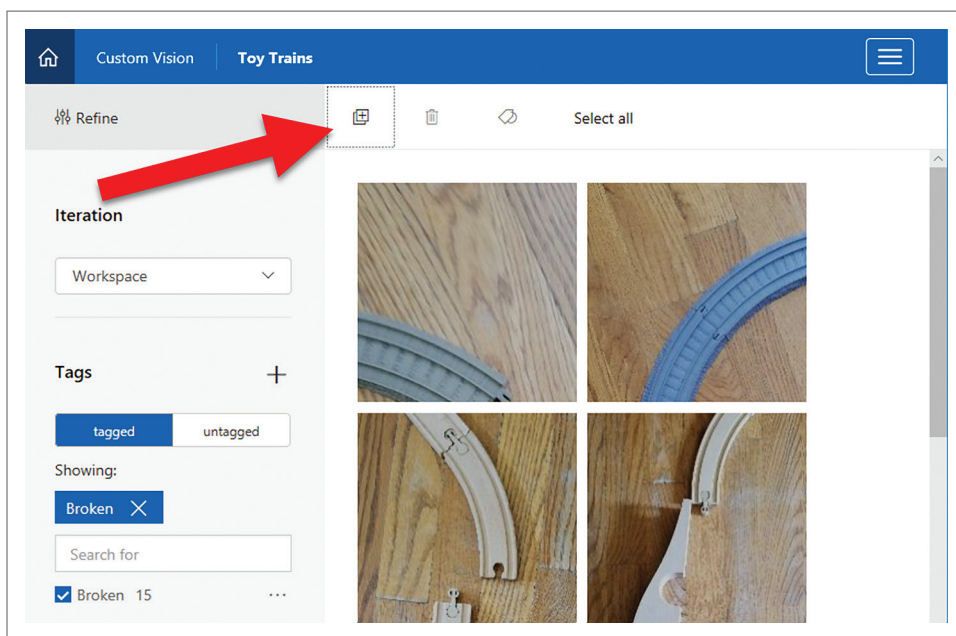


Figure 3 Adding More Images to the Training Data

Visual Studio **LIVE!**
EXPERT SOLUTIONS FOR .NET DEVELOPERS

September 17 – 20, 2018
Renaissance Chicago

Chicago

Look Back to
Code Forward.

#VSLIVE25

HELP US CELEBRATE 25 YEARS
OF CODING INNOVATION!



Visual Studio Live! (VSLive!™) is thrilled to be returning to Chicago this September where developers, software architects, engineers and designers will tackle training on the hottest topics (like .NET Core, Angular, VS2017), debate with industry and Microsoft insiders and network with your peers. Come experience the education, knowledge-share and networking at #VSLive25.

DEVELOPMENT TOPICS INCLUDE:



DevOps in the
Spotlight



Cloud, Containers
and Microservices



AI, Data and
Machine Learning



Developing New
Experiences



Delivery and Deployment



.NET Core and More



Full Stack Web Deployment

CONNECT WITH US



twitter.com/
vslive – @VSLive



facebook.com –
Search "VSLive"



linkedin.com – Join the
"Visual Studio Live" group!

Register by 8/17
& Save Up to \$200!
Use Promo Code MSDN



SUPPORTED BY



PRODUCED BY



vslive.com/chicago

Visual Studio **LIVE!**
EXPERT SOLUTIONS FOR .NET DEVELOPERS

October 7 – 11, 2018
Hilton San Diego Resort

San Diego

Code Again for
the First Time!

#VSLIVE25

DEVELOPMENT TOPICS INCLUDE:



DevOps in the
Spotlight



Cloud, Containers
and Microservices



AI, Data and
Machine Learning



Developing New
Experiences



Delivery and Deployment



.NET Core and More



Full Stack Web Deployment



Hands-On Labs

CODE WITH US IN SUNNY SAN DIEGO!



For the FIRST TIME EVER in our 25 year history, Visual Studio Live! is heading to San Diego, CA for up to 5 days of practical, unbiased, Developer training, including NEW intense hands-on labs. Join us as we dig into the latest features of Visual Studio 2017, including ASP.NET Core, Angular, Xamarin, UWP and more. Help us celebrate 25 years of coding innovation and experience the education, networking and knowledge-share at #VSLive25.

CONNECT WITH US



twitter.com/
vslive – @VSLive



facebook.com –
Search "VSLive"



linkedin.com – Join the
"Visual Studio Live" group!

Register by 8/3
for Best Savings!

Use Promo Code MSDN



SUPPORTED BY



PRODUCED BY



vslive.com/sandiego

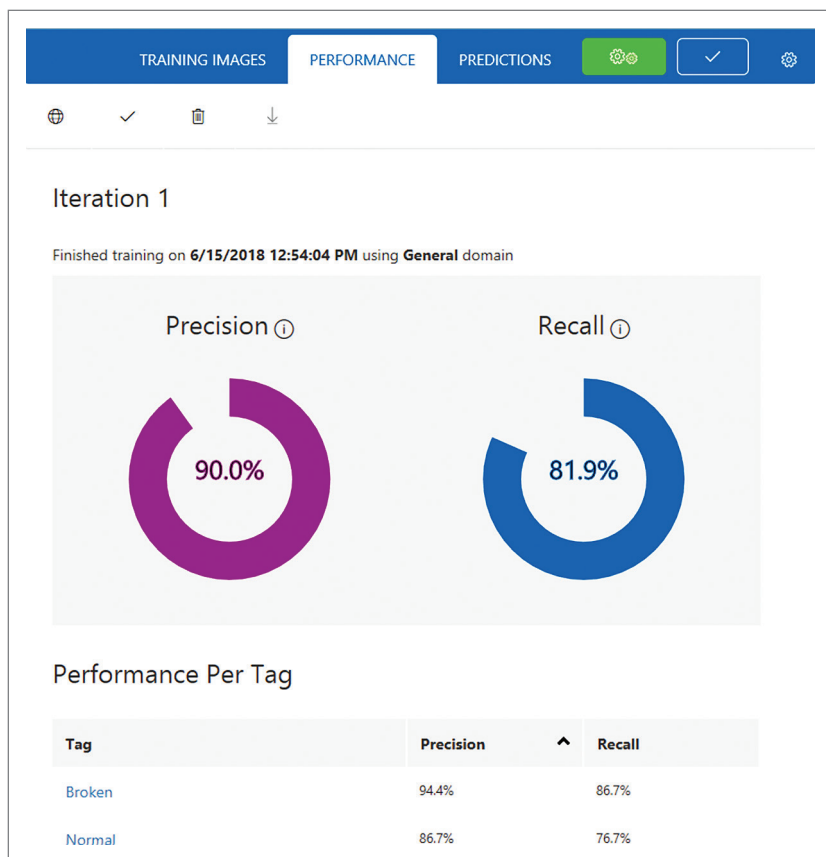


Figure 4 Training Results and Performance Metrics

the icon on the upper left of the pane with the plus sign on it. This button is highlighted in **Figure 3**. I repeat the previous steps to upload the input images, except this time, I choose the images in the Normal folder and tag the images with the label Normal. I click the Upload button to transfer the images and then click Done to close the dialog.

With all the sample images uploaded and tagged, it's time to train and test the model.

Training the Model

Toward the top of the page, there's a green button with gears on it. I click on this to train a model with the currently labeled images. It takes a moment or two before the training is complete and the Web page returns with the performance metrics of the model, which is shown in **Figure 4**.

The two primary metrics displayed are the model's precision and recall. Precision indicates how often a predicted result is correct, while recall measures the percentage of how often a predicted tag was correct. In other words, recall indicates that when the correct answer is "Broken,"

how often the model will predict "Broken." For a further explanation of Confusion Matrix terminology, read the "Simple Guide to Confusion Matrix Terminology" at bit.ly/2IbRoAi.

Testing the Model

Now it's time to test the model on the images set aside for testing to see how well the model will perform when presented with new images. To the immediate right of the green training button, there's a button labeled Quick Test. I click on this to bring up the testing dialog and then click the Browse local files button to bring up the file upload dialog. I use this dialog to select an image in the Test folder and click Open to upload the file. After a moment, the algorithm will display its results after evaluating the image. **Figure 5** shows that this image has a 79.2 percent probability of being a Normal track, and a 60.2 percent probability of being Broken. Remember that for this project, Normal means the track is unbroken, or contiguous.

I run the test again with another image of a broken track, by clicking again on the Browse local files button. This time I pick an image of a broken set of tracks. The model reports that there's a 95.9 percent chance of this being a broken track and only 11.2 percent chance of the track being normal. I test the model out with

the rest of the files in the Test folder to get an idea of where the model performs well and where it has trouble identifying the correct state of the tracks.

For further experimentation, I perform an image search on the Internet and copy the image URL and paste it into the text box on the Test UI. These images depict train tracks from different angles and on different surfaces in lighting conditions unlike the training set of images. This will provide a perspective on how neural

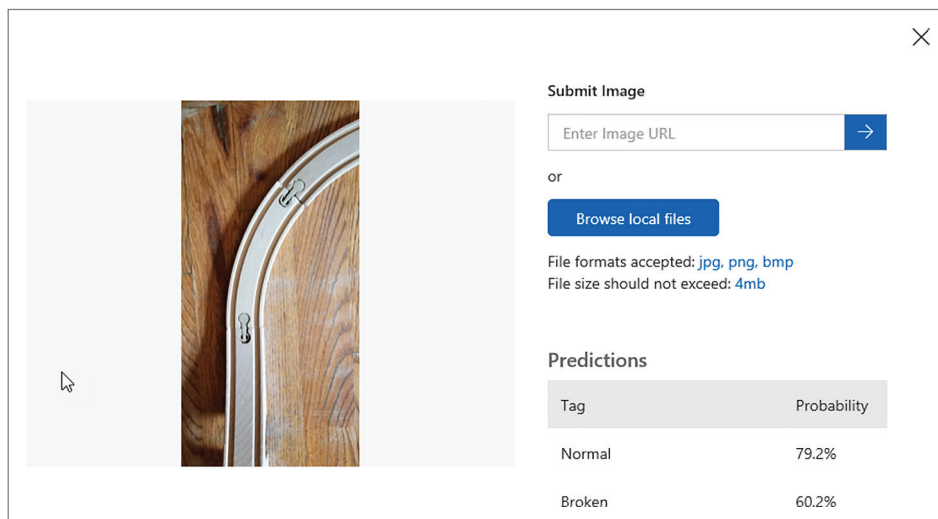


Figure 5 Quick Test Dialog for a Contiguous Set of Tracks

dtSearch®

Instantly Search Terabytes

dtSearch's **document filters** support:

- popular file types
- emails with multilevel attachments
- a wide variety of databases
- web data

Over 25 search options including:

- efficient multithreaded search
- **easy** **multicolor** **hit-highlighting**
- forensics options like credit card search

Developers:

- APIs for .NET, C++ and Java; ask about new cross-platform .NET Standard SDK with Xamarin and .NET Core
- SDKs for Windows, UWP, Linux, Mac, iOS in beta, Android in beta
- FAQs on faceted search, granular data classification, Azure and more

Visit dtSearch.com for

- hundreds of reviews and case studies
- fully-functional enterprise and developer evaluations

The Smart Choice for Text Retrieval®
since 1991

dtSearch.com 1-800-IT-FINDS

networks work and why a diverse set of sample data is crucial to the success of a production-ready model. The following three URLs point to some example images that I found online (note how the model classifies each of the images):

- bit.ly/2K8jd2x
- bit.ly/2IELsXH
- bit.ly/2NOGgdp

Furthermore, it's useful to know how the classifier will behave when given an image outside of its scope. For example, how will this classifier, designed to determine the state of toy train tracks, work when given an image of a neon sign? This is referred to as negative image handling, and an effective model ought to predict values close to zero in these instances. To test this out, I enter the following URL into the text box: <http://datadriven.tv/wp-content/uploads/2018/05/50k.png>. My predictive model produced a result of 1.7 percent for Normal and 0 percent for Broken.

Going Further

For purposes of this example, I created a two-class image classification system, where the model only had to label images of toy train tracks as either broken or normal. However, I can create more complex models. Currently, the Custom Vision Service S0 (Standard) tier can support up to 250 unique tags, meaning that it can be trained to classify 250 separate labels. Additionally, the service can handle up to 50,000 images per project.

Furthermore, the models generated by the Custom Vision Service can be exported for use on edge computing devices and do not rely on access to the cloud service. This means that I could load the model onto a device and classify images in an offline scenario. Currently, the Custom Vision Service supports model exports in three formats: TensorFlow for Android devices, CoreML for iOS devices and ONNX for Windows devices. Additionally, the Custom Vision Service can create a Windows or Linux container that incorporates a TensorFlow model and code to call the REST API. For more details on exporting models to run on edge devices, be sure to refer to the documentation at bit.ly/2K4ibjL.

In this article, I demonstrated how easy it is to get started with the Custom Vision Service and build out a computer vision model with a limited set of training data. One potential use for this technology would be to access image data from traffic cameras and train a model to detect different levels of congestion based solely on that data. Previously, automating such a task would be daunting, as it required both specialized knowledge and a massive amount of labeled training data. However, the Custom Vision Service combines cutting-edge neural network technology with an easy-to-use interface to create a tool that opens up machine vision to more widespread use. ■

FRANK LA VIGNE works at Microsoft as an AI Technology Solutions Professional where he helps companies achieve more by getting the most out of their data with analytics and AI. He also co-hosts the DataDriven podcast. He blogs regularly at FranksWorld.com and you can watch him on his YouTube channel, "Frank's World TV" (FranksWorld.TV).

THANKS to the following technical experts for reviewing this article:
Andy Leonard and Jonathan Wood



Artificial Intelligence LIVE!

AI FOR DEVELOPERS AND DATA SCIENTISTS

**ROYAL PACIFIC RESORT AT
UNIVERSAL ORLANDO
December 2-7, 2018**

A New AI Event for Developers and Data Scientists

Artificial Intelligence Live! is an innovative, new conference for current and aspiring developers, data scientists, and data engineers covering artificial intelligence (AI), machine learning, data science, Big Data analytics, IoT & streaming analytics, bots, and more. You can expect real-world training on the languages, libraries, APIs, tools and cloud services you need to implement real AI and machine learning solutions, today and into the future.

TRACK TOPICS INCLUDE:

- Artificial Intelligence (AI)
- Machine Learning
- Data Science
- Big Data analytics
- IoT & Streaming Analytics
- Bots
- & More....



**SAVE \$500 WITH SUMMER SAVINGS!
REGISTER BY AUGUST 31**

Use Promo Code MSDN

A Part of Live! 360: The Ultimate Education Destination

6 CONFERENCES, 1 GREAT PRICE

Visual Studio LIVE! | SQL Server LIVE! | **TECHMENTOR** | Artificial Intelligence LIVE! | Office & SharePoint LIVE! | Modern Apps LIVE!



ATTENDAILIVE.COM

EVENT PARTNERS



SILVER SPONSOR



PREMIER & ALLIANCE MEDIA PARTNERS



SUPPORTED BY



PRODUCED BY



What's New in ASP.NET Core 2.1

Steve Smith

Microsoft recently released ASP.NET Core 2.1 along with .NET Core 2.1 and Entity Framework (EF) Core 2.1. Combined, these releases offer some great improvements in performance, as well as additional features for .NET Core developers. Microsoft is also offering Long-Term Support (LTS) with this release, meaning it will remain supported for three years. This article provides an overview of the improvements in ASP.NET Core 2.1. To learn more about what's new in EF Core 2.1, check out this month's Data Points column by Julie Lerman, "Deep Dive into EF Core HasData Seeding" and her column last month (msdn.com/magazine/mt847184) that delves into the new EF Core 2.1 Query Type feature, which lets you more easily query a database without needing true entities with key properties to consume the results.

Razor Pages Improvements

I'll start by talking about improvements to Razor Pages, a new feature introduced in ASP.NET Core 2.0 that I wrote about in the September

2017 issue (msdn.com/magazine/mt842512). Version 2.1 adds a couple features that didn't make it into the 2.0 release, such as support for Pages-specific folders to search for shared assets. The most common shared assets are layout files and partials. By default, these were located in the root /Pages folder in ASP.NET Core 2.0, although ASP.NET Core MVC would discover them if they were placed in a /Views/Shared folder. In version 2.1, Razor Pages now searches for these shared files by looking for them in the following locations (in order):

1. The current Pages folder
2. /Pages/Shared
3. /Views/Shared

This allows you to easily override shared assets where desired, but if you're adding Razor Pages to an existing MVC-based application, you can continue to leverage any existing shared assets it has in its /Views/Shared folder.

Another feature that was missing when Razor Pages initially shipped was support for Areas. With ASP.NET Core 2.1, you can now add a /Pages folder with Razor Pages to any area located in the /Areas folder of your ASP.NET Core application. Microsoft has also updated the default Visual Studio Web Application template to use Areas for identity functionality when you choose "Individual user accounts."

Together, these features make it much easier to organize Razor Pages in the project system.

Shared Razor Libraries

Another new feature in 2.1 is support for loading Razor assets from separate libraries or packages. Separate ASP.NET Core apps

This article discusses:

- Razor Pages and shared Razor libraries
- New project templates with separate Identity add-on
- SignalR for ASP.NET Core

Technologies discussed:

ASP.NET Core, SignalR

Code download available at:

bit.ly/2JXdHeV

frequently share common assets, such as Identity features (login, register, forgot password and the like). Typically, these common features resulted in a lot of duplicate code across individual projects, leading to increased technical debt. The new Razor Class Library (RCL) feature supports building Razor files and deploying them as associated projects or NuGet packages that any number of ASP.NET Core apps can consume.

With the addition of this feature, the compiler will build Razor assets, automatically searching for related assets in referenced libraries and packages. Previously, Razor assets weren't built until after they were deployed and requested. ASP.NET Core 2.1 integrates Razor compilation into the build process, which also results in faster app start times.

Razor assets in RCLs can be overridden, so if you use them to share common assets between projects, you don't lose the ability to customize certain aspects on a per-project basis. RCLs shine for cross-cutting app concerns like layout, navigation and authentication. In fact, the built-in ASP.NET Core Identity feature is able to leverage this support to become more reusable between projects, as well.

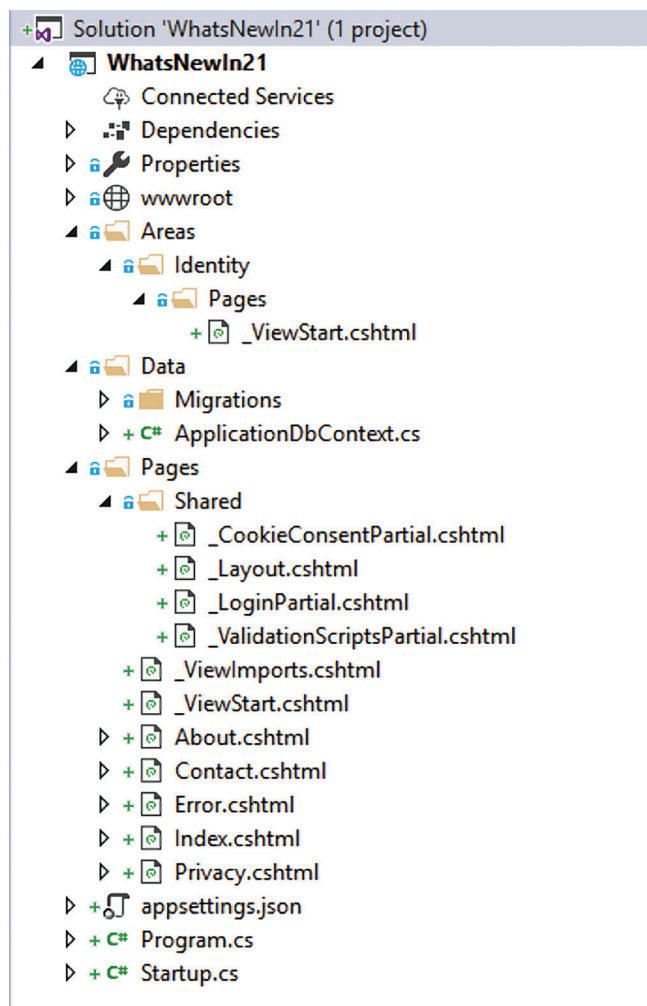


Figure 1 New Web Application Project Structure in ASP.NET Core 2.1

Project Template Updates

The default Web Application project template includes a few changes not previously mentioned. The New Project dialog now includes a checkbox to specify whether to enforce HTTPS. New support for General Data Protection Regulation (GDPR) is included by default, so projects now include default privacy pages and cookie consent Razor partials. Note that these documents are simply placeholders—it's your responsibility to add your organization's actual policy. **Figure 1** shows a brand-new ASP.NET Core 2.1 project, which has been configured to use Identity for individual user accounts.

Identity as an Add-On

If you wanted to take advantage of the ASP.NET Core Identity functionality in your app prior to version 2.1, you generally had to make the decision to add support for it when you were creating your Web app project. If you wanted to add Identity support later, typically the process would be to create a brand-new Web app project with the appropriate support (such as "Individual user accounts"), and then copy the files from the new app into your existing app. This was not an ideal solution.

It was always a tough technical challenge for ASP.NET Core (and ASP.NET, too) to support adding Identity support to an existing app, in part because this support included front-end Razor assets that couldn't be packaged separately, and deploying new assets into an existing app could fail for a host of reasons. With the addition of RCL support, adding Identity to an existing app becomes much easier, as the process no longer needs to add new Razor files to the existing app.

You can use scaffolding support in Visual Studio to add Identity to existing apps. New apps built with the latest templates will leverage shared Razor assets rather than including Identity-related pages/views in the project itself. The benefit of this approach is fewer boilerplate files in your new projects, but with no loss in functionality or your ability to customize behavior. You saw in **Figure 1** that by default Identity support only adds an Area with a single `_ViewStart.cshtml` file in it. You can customize the behavior of certain files by right-clicking on the project, clicking Add | New Scaffolded Item and then choosing Identity. Select the pages you want to scaffold and specify the DbContext class and any other options.

You'll see how I'm customizing these files when I get to SignalR.

Improved HTTPS Support

If you're not already using HTTPS for your Web apps, you probably should be. Search engines and browsers are now actively promoting sites that use HTTPS and treating those that don't as potentially insecure. GDPR requires that sites use HTTPS to protect user privacy. ASP.NET Core 2.1 bolsters support for developing and testing apps using HTTPS, including making it much easier to use HTTPS locally when building apps.

After you install the .NET Core 2.1 SDK, the first time you run it, it will install a development certificate. You can manage this certificate using the new `dotnet dev-certs` tool, which is installed with the SDK. Once the development certificate is trusted, you'll be able to develop and test your ASP.NET Core 2.1 apps locally using

HTTPS as your default protocol, more closely mirroring your production environment (where of course you're using HTTPS).

Your application can further improve its security by signaling to browsers that it supports HTTP Strict Transport Security (HSTS). HSTS helps prevent certain kinds of attacks, such as "man in the middle" attacks, by signaling to browsers that all responses for a given request must use HTTPS connections instead of plain HTTP. Without HSTS, even a page that's served via HTTPS might include resources that still use HTTP. These resources could easily be replaced or modified by routers in between the user and the servers hosting the content, because they're unprotected by encryption. HSTS prevents this attack vector.

HSTS is implemented with a response header in the HTTPS response of the original resource. For example:

```
Strict-Transport-Security: max-age=16070400; includeSubDomains
```

HSTS is enabled for your ASP.NET Core applications using middleware, configured by default in the application template's Startup.cs file. It's not recommended to use HSTS on localhost, so the default behavior only includes the middleware when the project is running in a production environment.

In addition to HSTS, you can also require HTTPS for your application using the new UseHttpsRedirection middleware. At its simplest, you enable this by adding the following line to your Configure method in Startup.cs:

```
App.UseHttpsRedirection();
```

Microsoft now recommends this for all ASP.NET Core apps, and it is the default in the Web application templates. This middleware will automatically redirect requests that come in over HTTP to HTTPS. It uses conventions to discover the appropriate HTTPS port, assuming only one is being used by the app. Alternately, you can configure by setting the ASPNETCORE_HTTPS_PORT environment variable (or http_port configuration key) or by specifying options in code in ConfigureServices:

```
services.AddHttpsRedirection(options => options.HttpsPort = 5555);
```

Updated SPA Templates

The application templates for Single Page Applications (SPAs) have been updated to use the latest recommended approaches for Angular and React apps. Specifically, the Angular template is now based on the Angular command-line interface (CLI), and the React templates are based on create-react-app (CRA). These SPA frameworks ship updates frequently, so updating the built-in templates to the latest approaches helps ensure new apps built with them will use current best practices for each associated framework.

SignalR

SignalR is a popular library that makes it very simple to add real-time Web functionality to ASP.NET applications. ASP.NET Core SignalR is a new version of SignalR that ships with ASP.NET Core 2.1. It features a number of improvements over previous versions:

- No client-side dependency on jQuery
- MessagePack-based binary protocol
- Based on Microsoft.AspNetCore.Sockets (not Http)
- Supports multiple formats (from same endpoint)

The server-side components of SignalR are included in Microsoft.AspNetCore.SignalR NuGet package. This package is included in the Microsoft.AspNetCore.App metapackage, so you typically should not need to add it separately to your ASP.NET Core project (assuming you're referencing Microsoft.AspNetCore.App version 2.1 or greater). SignalR supports multiple clients, including JavaScript for Web pages and a .NET client for .NET applications. The recommended way to add the JavaScript client to your project is through npm. Assuming you have npm installed, you can run the following commands to add the client to your project:

```
npm init -y
npm install @aspnet/signalr
```

In addition to HSTS, you can also require HTTPS for your application using the new UseHttpsRedirection middleware.

The first command initializes a packages.config for your project—you only need to run this command if you're not already using npm. The second command downloads the SignalR JavaScript client to your node_modules folder. You'll need to copy the signalr.js file from node_modules to an appropriate location in your ASP.NET Core app's wwwroot folder in order to reference it from your app.

SignalR Demo: Toast Notifications

To demonstrate how easy it is to get set up with SignalR, while also showing how to customize the behavior of the new Identity package, I've created a simple demo. It pops up a notification in the browser whenever a user registers or signs in or out of the app. This notification should appear anywhere on the site, so I'm going to modify the _Layout.cshtml file to include the client-side scripts necessary. Add the following to the bottom of the _Layout.cshtml file:

```
<script src="//cdnjs.cloudflare.com/ajax/libs/toastr.js/latest/js/toastr.min.js"></script>
<script src="~/lib/signalr/signalr.js"></script>
<script src="~/js/site.js" asp-append-version="true"></script>
```

The first reference is to a simple notification library called toastr that I'll use to display the notifications. The second is the signalr.js file copied from node_modules into the app's wwwroot/lib/signalr

Figure 2 Implementing the Logout PageModel OnPost Method

```
public async Task<IActionResult> OnPost(string returnUrl = null)
{
    string username = User.Identity.Name;
    await _signInManager.SignOutAsync();
    _logger.LogInformation("User logged out.");
    await _usersHubContext.Clients.All.SendAsync("ReceiveSignOut", username);
    if (returnUrl != null)
    {
        return LocalRedirect(returnUrl);
    }
    else
    {
        return Page();
    }
}
```




DevExpress DXperience 18.1 | from \$1,439.99



The complete range of DevExpress .NET controls and libraries for all major Microsoft platforms.

- WinForms - New TreeMap control, Chart series types and Unbound Data Source
- WPF - New Wizard control and Data Grid scrollbar annotations
- ASP.NET - New Vertical Grid control, additional Themes, Rich Editor Spell Checking and more
- Windows 10 Apps - New Hamburger Sub Menus, Splash Screen and Context Toolbar controls
- CodeRush - New debug visualizer expression map and code analysis diagnostics

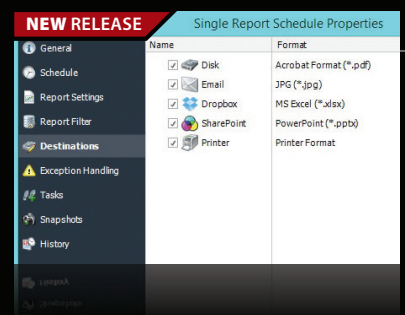


Help & Manual Professional | from \$586.04



Help and documentation for .NET and mobile applications.

- Powerful features in an easy, accessible and intuitive user interface
- As easy to use as a word processor, but with all the power of a true WYSIWYG XML editor
- Single source, multi-channel publishing with conditional and customized output features
- Output to responsive HTML, CHM, PDF, MS Word, ePub, Kindle or print
- Styles and Templates give you full design control

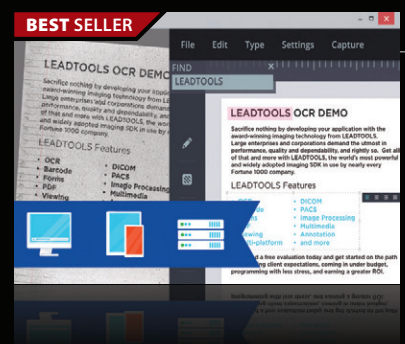


PBRS (Power BI Reports Scheduler) | from \$8,132.21



Data Driven Distribution for Power BI Reports & Dashboards.

- A comprehensive set of job (schedule) types gives you the power to automate delivery in Power BI
- Automate report delivery & send reports to printer, fax, folder, FTP, DropBox, SharePoint & email
- Contains powerful system event triggered, data-driven and business process workflow functions
- Respond instantly by firing off reports and automation scripts when an event occurs



LEADTOOLS Document Imaging SDKs V20 | from \$2,995.00 SRP



Add powerful document imaging functionality to desktop, tablet, mobile & web applications.

- Universal document viewer & conversion framework for PDF, Office, CAD, TIFF & more
- OCR, MICR, OMR, ICR and Forms Recognition supporting structured & unstructured forms
- PDF SDK with text edit, hyperlinks, bookmarks, digital signature, forms, metadata
- Barcode Detect, Read, Write for UPC, EAN, Code 39, Code 128, QR, Data Matrix, PDF417
- Zero-footprint HTML5/JavaScript UI Controls & Web Services

folder. The last is the existing site.js file that's created as part of a new ASP.NET Core project. The order of these scripts is important.

The next step is to modify site.js to add the JavaScript code necessary to display messages when a request is received from the server. There are three steps involved in this process. First, a connection must be created, using the `signalR.HubConnectionBuilder`. This type uses the builder design pattern to configure the connection with any necessary parameters, and then the connection is returned from the build method. Next, you configure message handlers using the `.on` method on the connection. Create one named handler for each behavior you want the server to be able to initiate. Finally, you initiate the connection by calling its start method:

```
const userUpdatesConnection = new signalR.HubConnectionBuilder()
    .withUrl("/userUpdates")
    .build();

userUpdatesConnection.on("ReceiveSignIn", (user) => {
    toastr.options.escapeHtml = true;
    const message = user + " logged in.";
    toastr.info(message, "New Login!");
});
// Additional handlers omitted
userUpdatesConnection.start().catch(err => console.error(err.toString()));
```

This code will run on every page that uses `_Layout.cshtml`. Now I'll configure the server end of the connection, starting in `Startup.cs`. You need to modify the `ConfigureServices` method to include `SignalR` (typically after calling `services.AddMvc()`):

```
services.AddSignalR();
```

Next, in the `Configure` method, you need to set up the `SignalR` routes, just before calling `app.UseMvc()`:

```
app.UseSignalR(routes =>
{
    routes.MapHub<UsersHub>("/userUpdates");
});
app.UseMvc();
```

Now it's time to add the `UsersHub` class that I referenced previously. You can put this class in a `Hubs` folder, which is a common approach, or you can think about using more of a feature-folder approach, as I described in my September 2016 article, "Feature Slices for ASP.NET Core MVC" (msdn.com/magazine/mt763233), and put the Hub with the Pages/Controllers with which it works. In any case, for this scenario, because the client isn't making calls to the hub, the class doesn't need any actual implementation. It just needs to inherit from `Microsoft.AspNetCore.SignalR.Hub`:

```
public class UsersHub : Hub
{
}
```

Finally, to use the Hub to communicate with connected clients from elsewhere in the app, you use dependency injection to inject `IHubContext<THub>` into the class that needs it. In this case, the PageModel classes for Login, Logout and Register each have an instance of `IHubContext<UsersHub>` injected into their constructors.

To send a message, use the `HubContext` instance to access its `Clients` property and send a message to a particular named handler.

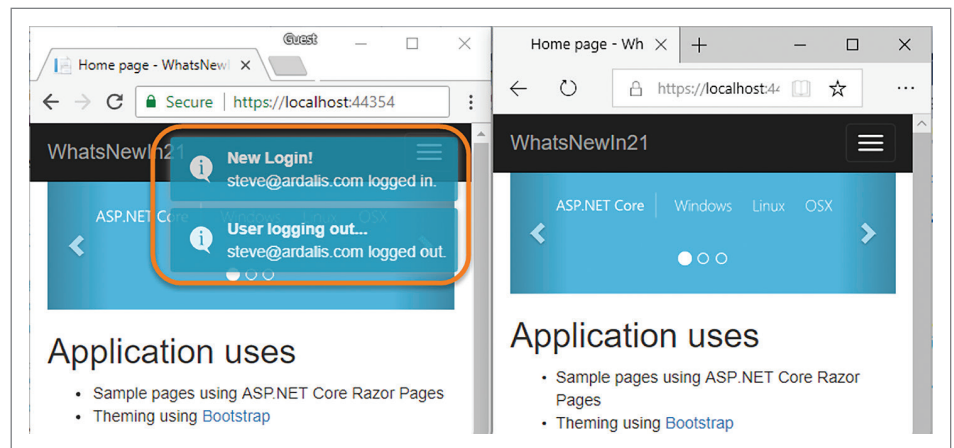


Figure 3 ASP.NET Core SignalR Sending Notifications from Server to Connected Browsers

Figure 2 shows the implementation for the Logout PageModel OnPost method.

With these pieces in place, you can run the application and open up several different browsers to display the app. In one, register and sign in and out of the app. You should see notifications appearing in the other browsers. Figure 3 demonstrates notifications appearing in a Chrome browser as a user signs in (on the left) and out of an Edge browser (on the right).

Integration Test Improvements

ASP.NET Core has had great support for integration testing the full stack in-memory since 1.0. However, one thing that required some custom setup to achieve was configuring the `TestServer` with the appropriate content root path, so that it could properly locate resources like views within the Web application. ASP.NET Core 2.1 introduces a new type, `WebApplicationFactory<T>`, which makes it easier to create a `TestServer` and an `HttpClient` that connects to it. To use the factory within an Xunit test class, you implement the `IClassFixture<WebApplicationFactory<Startup>>` interface, where `Startup` is the entry point for the ASP.NET Core app you wish to test. Then, in the class constructor, you inject a `WebApplicationFactory<Startup>` and use its `CreateClient` method to get an instance of a client. In your tests, you can then use the client to make requests to the application and verify that the correct response is returned, like so:

```
[Fact]
public async Task Get_HomePageReturnSuccessAndCorrectContentType()
{
    var response = await _client.GetAsync("/");
    response.EnsureStatusCode(); // Status Code 200-299
    Assert.Equal("text/html; charset=utf-8",
        response.Content.Headers.ContentType.ToString());
}
```

If you need to customize the application for testing, such as changing which services are used or adding seed data that will be used for testing, you can inherit from `WebApplicationFactory<T>`, and then use your custom factory in your tests.

Additional Improvements

This release also sees improvements to several other parts of ASP.NET Core 2.1, including the built-in Kestrel server. Kestrel now uses managed sockets for its default transport layer, instead

of libuv. This change should be seamless, but if it causes issues, developers can still configure libuv for use with Kestrel.

Another new addition to the hosting component of ASP.NET Core is that of the HostBuilder type, which you would use to configure non-Web parts of a host. HostBuilder is very similar to the existing WebHostBuilder, but doesn't allow you to specify a Startup class from a Web project. It's designed to let you configure common concerns like dependency injection, configuration, and logging for non-Web scenarios like hosted services or console applications.

ASP.NET Core has had
great support for integration
testing the full stack
in-memory since 1.0.

Finally, if you're writing API endpoints in your ASP.NET Core app, you can take advantage of some improvements added in 2.1. First, you can add the [ApiController] attribute to any of your controllers that expose APIs. This adds a number of features to endpoints defined on these controllers, such as:

- Model validation errors will automatically return BadRequest(ModelState)
- Binding sources (such as [FromBody]) automatically inferred for action parameters
- File uploads using [FromForm] automatically infer multipart/form-data content type
- Attribute routing is required

Another addition is a new return type, ActionResult<T>. This return type is used in place of IActionResult and allows type information to be included in the method signature. Tools like Swashbuckle can use this information to generate OpenAPI/Swagger documentation. Without this return type, you would need to annotate methods that simply returned IActionResult with the [ProducesResponseType] attribute to expose this information.

Next Steps

If you're not already using ASP.NET Core 2.1, I recommend upgrading as soon as possible. You can get the SDK at microsoft.com/net/download/windows or run the tools using Docker from one of the images at dockr.ly/2MAaiEF. The updated source code for this sample is available at bit.ly/2JXdHeV. ■

STEVE SMITH is an independent trainer, mentor and consultant. He blogs at ardalis.com and provides developer tips via e-mail and podcast at WeeklyDevTips.com. Check out his courses on Pluralsight to learn how to develop cleaner, higher quality applications. Follow him on Twitter: @ardalis.



SUPER-FAST AND ADVANCED CHARTS

LightningChart®

- WPF and WinForms
- Real-time scrolling up to 2 billion points in 2D
- Hundreds of examples
- On-line and off-line maps
- Advanced Polar and Smith charts
- Outstanding customer support



2D charts - 3D charts - Maps - Volume rendering - Gauges
www.LightningChart.com/ms

TRY FOR
FREE



Visual Studio® LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS

September 17-20, 2018
Renaissance

Chicago

Visual Studio LIVE
25
YEARS OF CODING INNOVATION
1993 - 2018

Look Back to Code Forward

Visual Studio Live! (VSLive!™) is thrilled to be returning to Chicago where developers, software architects, engineers and designers will “look back to code forward” during four days of unbiased and cutting-edge education on the Microsoft Platform.

Tackle training on the hottest topics (like .NET Core, Angular, VS2017), debate with industry and Microsoft insiders (people like Rockford Lhotka, Deborah Kurata and Brock Allen) and network with your peers—plus, help us celebrate 25 years of coding innovation as we take a fun look back at technology and training since 1993. Come experience the education, knowledge-share and networking at #VSLive25.

DEVELOPMENT TOPICS INCLUDE:



**DevOps in the
Spotlight**



**Cloud, Containers
and Microservices**



**AI, Data and
Machine Learning**



**Developing New
Experiences**



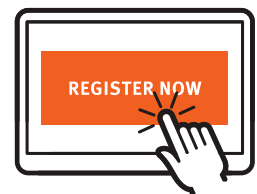
**Delivery and
Deployment**



**.NET Core
and More**



**Full Stack
Web Development**



vslive.com/chicago

**Save Up To \$300
When You Register
by August 17**

Use Promo Code MSDN

GOLD SPONSOR



SILVER SPONSOR



SUPPORTED BY



PRODUCED BY



DevOps in the Spotlight		Cloud, Containers and Microservices		AI, Data and Machine Learning		Developing New Experiences		Delivery and Deployment		.NET and More		Full Stack Web Development	
START TIME	END TIME	Pre-Conference Workshops: Monday, September 17, 2018 <small>(Separate entry fee required)</small>											
8:00 AM	9:00 AM	Pre-Conference Workshop Registration - Coffee and Morning Pastries											
9:00 AM	6:00 PM	M01 Workshop: Build a Modern ASP.NET App in the Cloud with a full CI/CD Pipeline in VSTS - <i>Brian Randell</i>				M02 Workshop: SQL Server for Developers - <i>Andrew Brust and Leonard Lobel</i>				M03 Workshop: Distributed Cross-Platform Application Architecture - <i>Rockford Lhotka and Jason Bock</i>			
6:45 PM	9:00 PM	Dine-A-Round											
START TIME	END TIME	Day 1: Tuesday, September 18, 2018											
7:00 AM	8:00 AM	Registration - Coffee and Morning Pastries											
8:00 AM	9:15 AM	T01 The State of XAML: Recent Changes for UWP, WPF, Xamarin - <i>Billy Hollis</i>			T02 An Introduction to TypeScript - <i>Jason Bock</i>			T03 SQL Server Security Features for Developers - <i>Leonard Lobel</i>			T04 Get Started with Git - <i>Robert Green</i>		
9:30 AM	10:45 AM	T05 Building Your First Mobile App with Xamarin Forms - <i>Robert Green</i>			T06 Essential Web Development with ASP.NET Core - <i>Mark Michaelis</i>			T07 Exploring T-SQL Enhancements: Windowing and More - <i>Leonard Lobel</i>			T08 DevOps for the SQL Server Database - <i>Brian Randell</i>		
11:00 AM	12:00 PM	KEYNOTE: To Be Announced - <i>Amanda Silver, Partner Director of Program Management, Microsoft</i>											
12:00 PM	1:30 PM	Lunch											
1:30 PM	2:45 PM	T09 Cross-Platform App Dev with C# and CSLA .NET - <i>Rockford Lhotka</i>			T10 Assembling the Web—A Tour of WebAssembly - <i>Jason Bock</i>			T11 Glue for the Internet: Introducing Azure Event Grid - <i>Jeremy Likness</i>			T12 Azure DevOps with VSTS, Docker, and K8 - <i>Brian Randell</i>		
3:00 PM	4:15 PM	T13 A Dozen Ways to Mess Up Your Transition From Windows Forms to XAML - <i>Billy Hollis</i>			T14 Entity Framework Core 2 For Mere Mortals - <i>Philip Japikse</i>			T15 Code First in the Cloud: Serverless .NET with Azure - <i>Jeremy Likness</i>			T16 Essential C# 8.0 - <i>Mark Michaelis</i>		
4:15 PM	5:30 PM	Welcome Reception											
START TIME	END TIME	Day 2: Wednesday, September 19, 2018											
7:30 AM	8:00 AM	Registration - Coffee and Morning Pastries											
8:00 AM	9:15 AM	W01 Electron: Desktop Development For Web Developers - <i>Chris Woodruff</i>			W02 JavaScript for the C# (and Java) Developer - <i>Philip Japikse</i>			W03 Quantum Computing and the Future of Software Development - <i>Jerry Nixon</i>			W04 Building a Stronger Team, One Strength at a Time - <i>Angela Dugan</i>		
9:30 AM	10:45 AM	W05 Enhancing UWP Experiences with Fluent Design - <i>Tony Champion</i>			W06 Architecting and Developing Microservices Apps - <i>Eric D. Boyd</i>			W07 Sharing C# Code Across Platforms - <i>Rockford Lhotka</i>			W08 How do You Measure up? Collect the Right Metrics for the Right Reasons - <i>Angela Dugan</i>		
11:00 AM	12:00 PM	General Session: To Be Announced - <i>Jay Schmelzer, Director of Program Managemnet, Visual Studio Team Microsoft</i>											
12:00 PM	1:00 PM	Birds-of-a-Feather Lunch											
1:00 PM	1:30 PM	Dessert Break - Visit Exhibitors - Exhibitor Raffle @ 1:15pm (Must be present to win)											
1:30 PM	2:45 PM	W09 Learning The Language Of HTTP For A Better Data Experience In Your Mobile Apps - <i>Chris Woodruff</i>			W10 Angular 101 - <i>Deborah Kurata</i>			W11 Power BI: What Have You Done for Me Lately? - <i>Andrew Brust</i>			W12 Fault Driven Development - <i>Josh Garverick</i>		
3:00 PM	4:15 PM	W13 Building Cross Device Experiences with Project Rome - <i>Tony Champion</i>			W14 N Things You Didn't Know About the Router - <i>Deborah Kurata</i>			W15 Analytics and AI with Azure Databricks - <i>Andrew Brust</i>			W16 Core Azure Solutions: Automation - <i>Josh Garverick</i>		
4:30 PM	5:45 PM	W17 Use UWP to Modernize Your Existing WinForms and WPF Applications - <i>Walt Ritscher</i>			W18 Tools for Modern Web Development - <i>Ben Hoelting</i>			W19 Create Intelligent Bots with Cognitive Services and Azure Search - <i>Eric D. Boyd</i>			W20 Real World Scrum with Team Foundation Server & Visual Studio Team Services - <i>Benjamin Day</i>		
6:30 PM	9:00 PM	VSLive!'s Windy City Sunset Cruise											
START TIME	END TIME	Day 3: Thursday, September 20, 2018											
7:30 AM	8:00 AM	Registration - Coffee and Morning Pastries											
8:00 AM	9:15 AM	TH01 PowerShell for Developers - <i>Brian Randell</i>			TH02 Docker for ASP.NET Core Developers - <i>Michele Leroux Bustamante</i>			TH03 MVVM and ASP.NET Core Razor Pages - <i>Ben Hoelting</i>			TH04 Unit Testing & Test-Driven Development (TDD) for Mere Mortals - <i>Benjamin Day</i>		
9:30 AM	10:45 AM	TH05 From Waterfall to Agile. Microsoft's Not-So-Easy Evolution into the World of DevOps - <i>Abel Wang</i>			TH06 Developing Microservices Solutions on Azure - <i>Michele Leroux Bustamante</i>			TH07 Eliminate Code Using Data Binding in WPF - <i>Paul Sheriff</i>			TH08 C# 7, Roslyn and You - <i>Jim Wooley</i>		
11:00 AM	12:15 PM	TH09 Writing Testable Code and Resolving Dependencies—DI Kills Two Birds with One Stone - <i>Miguel Castro</i>			TH10 Effective Data Visualization - <i>David Giard</i>			TH11 Store Data Locally for Offline Web Applications - <i>Paul Sheriff</i>			TH12 Improving Code Quality with Static Analyzers - <i>Jim Wooley</i>		
12:15 PM	1:15 PM	Lunch											
1:15 PM	2:30 PM	TH13 Exposing an Extensibility API for Your Applications and Services - <i>Miguel Castro</i>			TH14 Adding Image and Voice Intelligence to Your Apps with Microsoft Cognitive Services - <i>David Giard</i>			TH15 Modern Security Architecture for ASP.NET Core - <i>Brock Allen</i>			TH16 SQL Server 2017—Intelligence Built-in - <i>Scott Klein</i>		
2:45 PM	4:00 PM	TH17 Advanced DevOps—Deep Dive into Feature Flags - <i>Abel Wang</i>			TH18 Programming with Microsoft Flow - <i>Walt Ritscher</i>			TH19 Implementing Authorization in Web Applications and APIs - <i>Brock Allen</i>			TH20 Databases and Data Lakes—Bridging the Gap - <i>Scott Klein</i>		

Speakers and sessions subject to change

CONNECT WITH US



twitter.com/vslive – @VSLive



facebook.com – Search "VSLive"



linkedin.com – Join the "Visual Studio Live" group!

vslive.com/chicago

Blockchain Fundamentals: Diving into Transaction Hash Chains

Jonathan Waldman

In the first article in this series (msdn.com/magazine/mt845650), I presented foundational concepts required to broadly understand modern-day blockchains, using high-level examples to illustrate the basics. In this article, I'll revisit some of the topics from the previous article by going into more detail about transaction hash chains, the role of the transaction pool and how a longest blockchain always prevails. This article is best read as a supplement to the previous article and contains introductory material for developers new to blockchain technologies.

Incidentally, while the articles in this series are based on the Bitcoin blockchain, I'm not advocating the adoption of a particular blockchain product or technology. Rather, my goal is to explore the foundation on which popular blockchain technologies

are built and to equip you with knowledge you can apply should you decide to leverage existing blockchains or engineer your own. As you study blockchains, you'll soon realize that implementation details differ dramatically among them. If you decide to specialize in a particular blockchain implementation, you'll need to keep pace with its fixes and updates in order to maintain expertise. But I warn you that the dynamic nature of these emerging technologies often means that available books, videos, blogs, forums and other documentation resources fall behind, sometimes making it necessary to consult the latest-deployed source code as a particular blockchain implementation's definitive reference.

Transaction Hash Chain Revisited

My previous article discussed the transaction hash chain data structure, which tracks digital asset ownership. In this article, I'll delve more deeply into how that hash chain works.

To pay homage to blockchain's roots, I'll begin by focusing on Satoshi Nakamoto's seminal white paper about Bitcoin (bitcoin.org/bitcoin.pdf) published on Oct. 31, 2008—months before the launch of Bitcoin on Jan. 3, 2009. Although Bitcoin implementation details have changed quite a bit since then, the white paper remains a useful reference, in particular the diagram on p. 2 that expresses the original transaction hash chain design concept.

The purpose of that diagram is to convey how a transaction hash chain is constructed and how digital signatures authorize the transfer of ownership sequence. However, it's highly abstracted

This article discusses:

- Transaction hash chain implementation details
- How digital assets are signed and transferred to a new owner
- The role of the transaction pool
- Why consensus algorithms are needed
- Proof-of-work versus proof-of-stake
- Why a longest chain always prevails

Technologies discussed:

Blockchain, Transaction Hash Chain, Elliptic Curve Cryptography, Proof-of-Work Algorithm, Proof-of-Stake Algorithm



Your Next Great App Starts Here

Q&A with Julian Bucknall,
Chief Technical Officer, DevExpress

Q DevExpress is celebrating its 20th year in the software development tools industry. What's been the secret to success at DevExpress over these twenty years?

A I believe the secret to our success has been our loyal customer base. Over the years, they've pushed us to deliver the very best possible tools and have made certain to hold our feet to the fire when we've fallen short. Suffice it to say, we'd be nothing without our wonderful users—they've made us who we are.

Q What type of challenges do you think today's software developers face and how are you addressing these challenges?

A Well, the software industry is highly fragmented. Application boundaries have shifted, and software consumers expect each and every solution to be accessible across a broad range of computing devices. Our job is to listen to our customers, evolve our product line, and address their business needs with tools that work as expected. Developers don't have time to waste and they certainly don't have time for excuses. They need products that deliver on promises made.

Q A fragmented tech landscape suggests the need to support many different platforms simultaneously. What would you say are the platforms of highest importance to you and your customers?

A We currently ship over 500 individual products targeting various developer platforms, from WinForms and WPF to ASP.NET, .NET Core and Angular. I'd be lying if I told you we support everything for everyone. No single organization can make such a claim. What I can promise to our users is that we are fully committed to Visual Studio, the .NET Framework and next-gen

platforms like .NET Core. Heck, we're even working on tools for Visual Studio Code.

Q What reason would you give a Visual Studio developer to choose DevExpress for their next software project?

A One way to judge the merits of our product line is to count the number of Visual Studio Readers' Choice Awards we've received over the years. The award list is quite long. Ultimately, however, I believe that each developer must decide for themselves. It's not good enough for me to tell a dev how great we are. A developer ought to make that determination for themselves. If you're in the market for UI components, a reporting engine, an analytics dashboard or the like, download our free trial, compare our products to the competition and decide. I think you'll be pleasantly surprised by what you discover when you start using DevExpress tools.



To access your 30-day, risk-free trial visit →

www.devexpress.com/trial

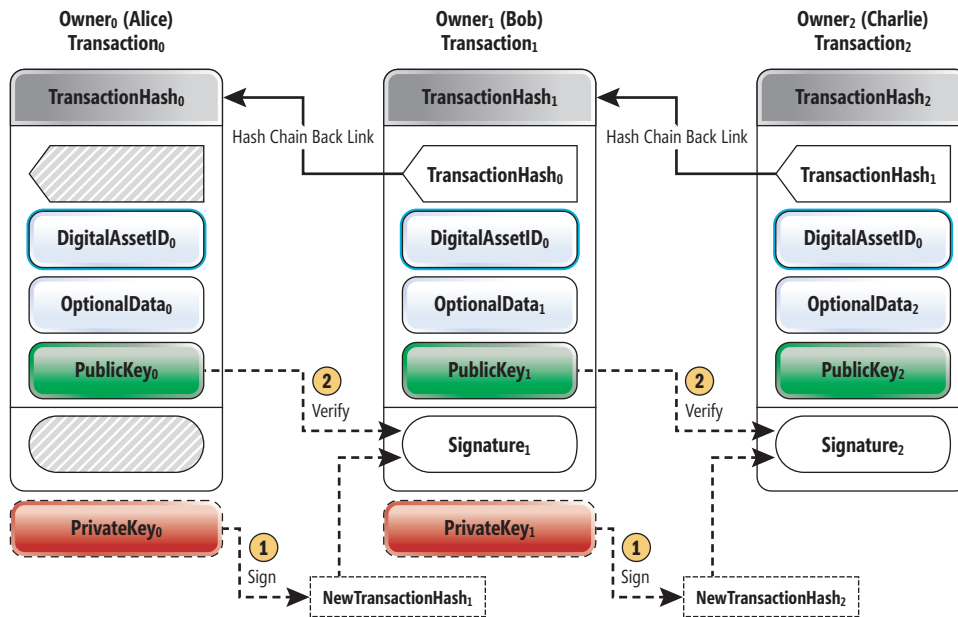


Figure 1 Updated Version of Satoshi Nakamoto's Original Transaction Hash Chain Diagram

and, as such, is a bit confusing. To add clarity, I've constructed a more detailed version that depicts how current transaction hash chains actually work (see Figure 1).

My revised diagram shows three transactions (0-based, as in the original document): Transaction₀ for Alice, Transaction₁ for Bob, and Transaction₂ for Charlie. The first transaction establishes Alice the original owner of the digital asset; its second transaction transfers ownership to Bob; and its third transaction transfers ownership to Charlie. Each transaction is composed of these fields (shown with a solid outline): transaction hash, digital asset ID, optional data, public key and signature. Other fields are used but not stored in the transaction (shown with a dashed outline): private key and new transaction hash. The diagram expresses field values as subscripted, mixed-case names—for example, the transaction hash value for Transaction₀ is TransactionHash₀ and the public key value for Transaction₂ is PublicKey₂.

As you study blockchains,
you'll soon realize that
implementation details differ
dramatically among them.

Figure 1 is a simplified transaction hash chain because it tracks only a single digital asset (DigitalAssetID₀) as it changes ownership (in contrast, cryptocurrency transaction hash chains typically have multiple digital asset inputs and outputs). Also, don't confuse the transaction hash chain with the blockchain, which aggregates verified transactions into blocks. Finally, the transaction hash chain isn't typically stored as the single linked-list data structure

depicted. Rather, it can be constructed (quickly, with the aid of indexes) from transaction data that's stored on the blockchain.

As I described in my previous article, the sequence of transactions is preserved because each new owner's transaction contains a hash value that's back-linked to the previous owner's transaction. In Figure 1, back-links are formed when the transaction hash of the previous transaction is stored in the current transaction. For example, Bob's transaction contains a transaction hash field containing Alice's TransactionHash₀ value; likewise, Charlie's transaction contains a transaction hash field containing Bob's TransactionHash₁ value, and so on.

Back-links are just one of several data-integrity components of the transaction hash chain. The chain also enforces transfer-of-ownership authorization. To follow an example, imagine that Alice is a purveyor of the world's finest wines and wants to maintain a ledger that tracks the fate of every bottle she owns. One day, Alice goes to her wine cellar and decides she will register herself on her business's blockchain as the original owner of every bottle of wine stocked there, effectively seeding transaction hash chains for each of her cherished bottles of wine. To begin, she casually grabs a bottle of Cheval Blanc 1947 Saint-Émilion and tags it with a QR code containing a unique ID. She then scans the QR label into her blockchain client software running as a node on the network. The software translates the scanned code into a digital asset ID (DigitalAssetID₀) then adds optional data (OptionalData₀) along with Alice's public key (PublicKey₀). As you can see in Figure 1, these fields are in their own outlined rectangle that represents an unsigned transaction. Each transaction also contains a transaction hash back-link and a signature, but because this is the first transaction in the hash chain, those fields are blank (shown by the shaded fields for Transaction₀).

Shown atop each transaction is a unique transaction hash value that the client software calculates by SHA-256-hashing together all of the transaction fields (the transaction hash, digital asset ID, optional data, owner's public key and signature). Again, it's this transaction hash value that's used as the next transaction's back-link for DigitalAssetID₀.

When Bob, the manager of Alice's Manhattan restaurant, wants to acquire Alice's bottle of Cheval Blanc, he uses his client software to generate a new public-private key pair for the transaction. Bob could skip this step and aggregate all of his digital assets under a single, previously used public key, but that would expose him to unnecessary risk. Instead, he generates a new key pair and gives Alice a public key he's never used before. That way, if he ever loses the paired private key, he loses access to only a single digital asset.

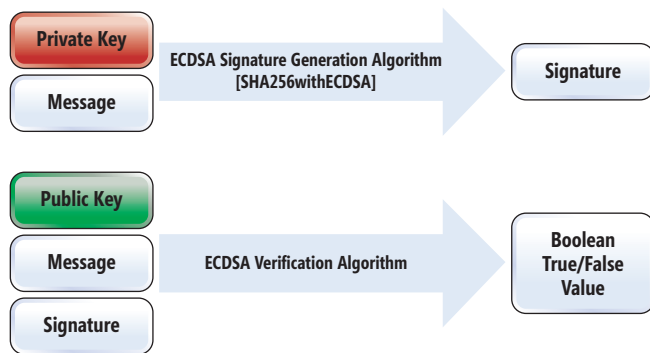


Figure 2 Elliptic Curve Digital Signature Algorithm Signature Generation (Top) and Verification Algorithm (Bottom)

In response to Bob's request, Alice launches her client software and browses her digital assets. She selects the transaction ID associated with the Cheval Blanc bottle Bob wants and then initiates the transfer request by providing Bob's public key, which doubles as a sort of destination address. The node then creates a new transaction (Transaction_1) containing the back-link value of the previous transaction hash (TransactionHash_0), the value of the digital asset ID (DigitalAssetID_0) for the Cheval Blanc bottle (this is the same value as the digital asset ID for Transaction_0), the value for any custom fields related to the transaction (OptionalData_1), and the value of Bob's public key (PublicKey_1) because Bob is this transaction's owner.

So far, the node has constructed an unsigned new Transaction_1 for Bob. The next step is to sign the transaction using Alice's private key. This is a critical step: Alice currently owns the digital asset in question so only she can authorize transfer of that digital asset to Bob.

Elliptic Curve Cryptography

In **Figure 1**, labels 1 and 2 indicate where the transaction is signed and where it's verified, respectively. In its current version, the Bitcoin blockchain leverages an implementation of public key cryptography (PKC) called elliptic curve cryptography (ECC). ECC provides stronger cryptographic results and shorter keys than the popular RSA/Diffie-Hellman alternative. Blockchain nodes use ECC to generate asymmetric key pairs using a formula that involves randomly selected points on a two-dimensional graph. This scheme allows a lost public key to be regenerated from the private key (but of course doesn't allow a lost private key to be regenerated from a public key).

Blockchains modeled after Bitcoin also leverage ECC when it comes to digital signatures. Unlike the simplified PKC Rivest-Shamir-Adelman (RSA) examples I illustrated in my previous article, Bitcoin now uses an Elliptic Curve Digital Signature Algorithm (ECDSA) (specifically, SHA256withECDSA) for signing transactions. This algorithm works a little differently from other signing technologies: In ECDSA, you must pass the signer's private key along with the message to be signed to a function that uses an ECDSA signature-generation algorithm to create a signature (this step is indicated by marker 1 in **Figure 1**). To later verify that signature, you must pass the signer's public key, message, and signature

to a function that uses an ECDSA verification algorithm to generate a true or false value, indicating whether or not the signature is valid (this step is indicated by marker 2 in **Figure 1**). **Figure 2** summarizes signing and verifying using ECDSA.

When creating a digital signature using PKC RSA, you verify the signature by comparing hash values, as shown in my previous article. For the curious-minded, that signature verification strategy isn't possible with ECDSA. RSA PKC is a deterministic digital signature algorithm because signing a given message with a given private key generates the same signature every time. ECDSA, on the other hand, is non-deterministic: Each time you pass a message and a private key to the ECDSA signing function you'll obtain a different signature. To see this in action, go to bit.ly/2MCTuwl.

Continuing the example, Alice is about to sign the transaction that transfers ownership of DigitalAsset_0 to Bob. The node software passes her private key (PrivateKey_0) along with a message ($\text{NewTransactionHash}_1$) to the ECDSA signature-generation algorithm function and obtains a signature as output (Signature_1). The node adds this signature value to the signature field of the new transaction. Finally, the node calculates the transaction hash (TransactionHash_1) value, which is a SHA-256 hash of all transaction fields, including the signature. At that point, the node successfully produced a signed transaction that can be sent to the transaction pool.

A signed transaction is deemed unverified until it has been validated by a miner node. When a miner node tries to verify Bob's transaction, it uses the transaction hash back-link to access the previous transaction's public key, which leads to Alice's Transaction_0 . Once the node has access to the previous transaction, it passes that transaction's public key (PublicKey_0) along with the new transaction hash ($\text{NewTransactionHash}_1$) and the signature in Bob's transaction (Signature_1) to the ECDSA verification algorithm that returns a true or false value, indicating whether or not the signature is valid.

A signed transaction is deemed
unverified until it has been
validated by a miner node.

Incidentally, Alice's private key (PrivateKey_0) and the new transaction hash ($\text{NewTransactionHash}_1$) are not stored in the transaction. Private key values should not be stored on a blockchain, and there's no need to store the new transaction hash value because it can be recomputed whenever needed.

Bob grabs his corkscrew and thinks he's going to savor the Cheval Blanc when he receives a Skype call from Charlie, the manager of one of Alice's other restaurants. Charlie wants to offer a special bottle of wine to welcome a newly hired sommelier. Bob regretfully agrees to transfer ownership of the Cheval Blanc to Charlie. He asks for Charlie's public key, and the same process is again carried out in order to transfer DigitalAsset_0 ownership from Bob to Charlie.

There now exist three transactions for DigitalAsset_0 —one for Alice, one for Bob and one for Charlie. Each transaction was

TEXTCONTROL

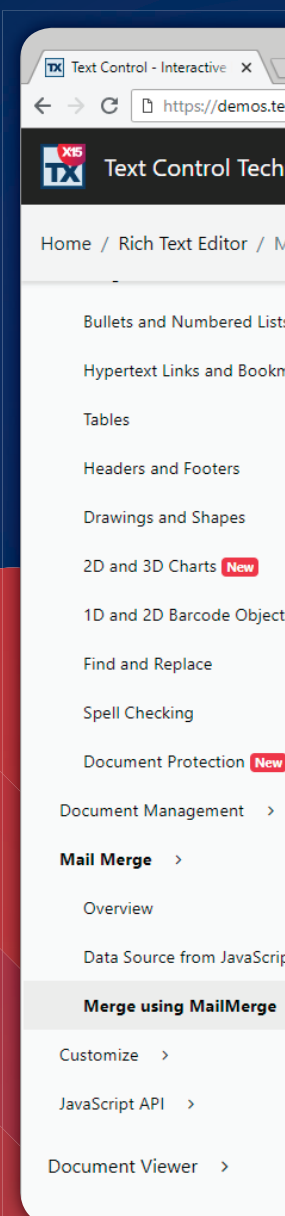
Integrate Documents and Reporting in Any Platform

The Text Control Reporting Framework combines powerful reporting features with an easy-to-use, MS Word compatible, word processor.

See our technology live:

demos.textcontrol.com

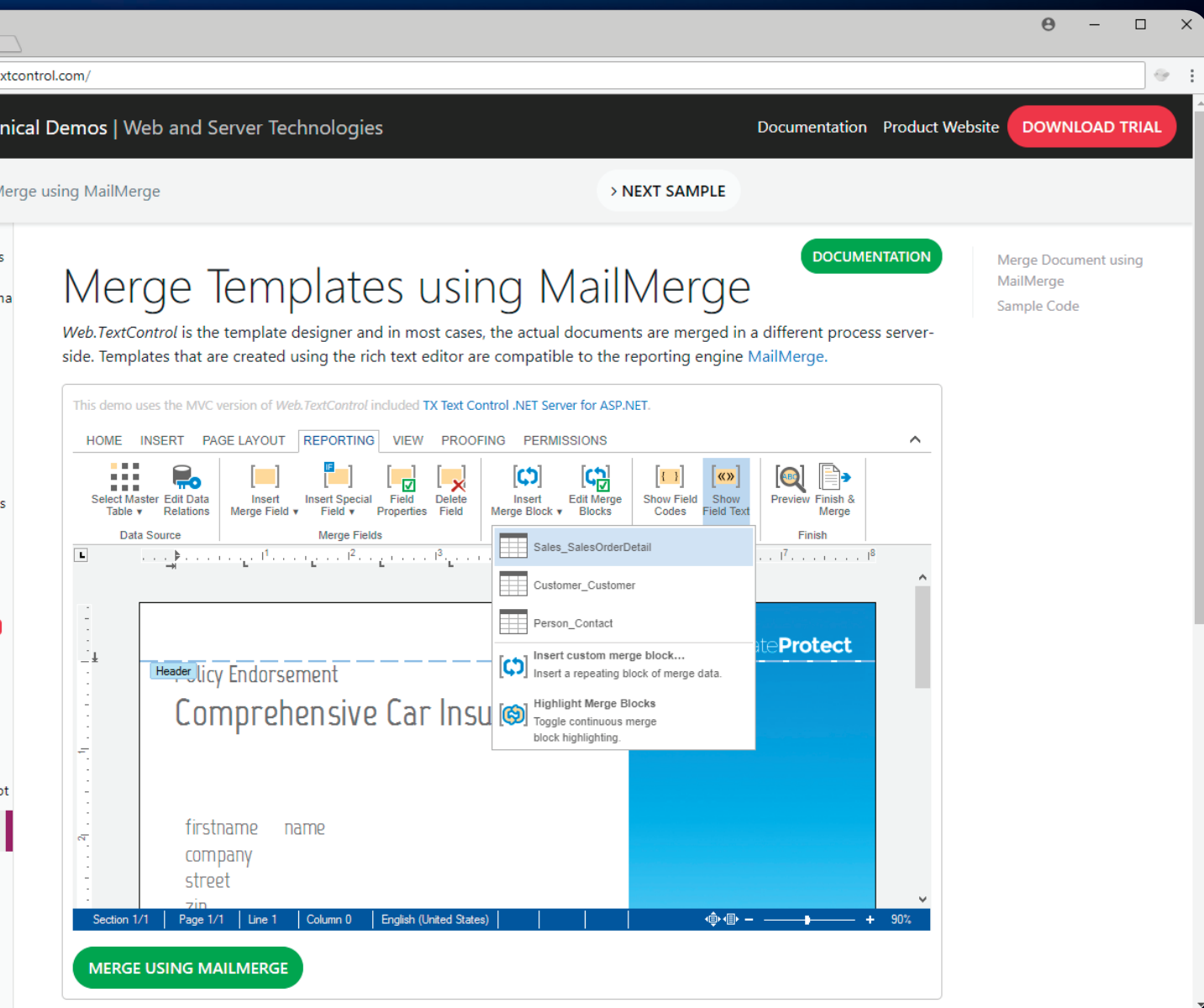
**WE ARE CHANGING
THE WAY YOU LOOK AT
REPORTING**



SEE THIS LIVE

TX Text Control X15 Technical Demos

Evaluate our technology and test the most sophisticated, cross-browser, and true WYSIWYG, rich text editor. Merge MS Word compatible templates with JSON data and create pixel-perfect Adobe PDF documents on-the-fly. See what's possible today!



verified and incorporated into the blockchain. After a certain number of additional blocks have been mined on top of the block that contains a particular transaction, that transaction is deemed confirmed (this “certain number” is implementation-specific). Thus, the official owner of a particular digital asset is always the person who has the private key to the most recently confirmed transaction for that digital asset’s transaction hash chain.

The Need for Consensus

As you’ve seen, a transaction hash chain is a data structure that strives to enforce ownership of a digital asset. But remember that these transactions are stored on a distributed, decentralized, asynchronous, public network that’s vulnerable to attacks and exposed to nodes that don’t necessarily honor blockchain protocol rules (so-called “bad actors”). The result is that bad-actor nodes could verify transactions that aren’t actually valid or could collude on the network to undermine the integrity of the blockchain.

The Transaction Pool To prevent these transaction-integrity issues, all transactions go through a verification and confirmation process. Each transaction is created by a single node somewhere on the network. For example, assume that Alice is in Albuquerque and Bob is in Boston. When Alice transfers ownership of her digital asset to Bob, Transaction₁ is constructed by a node in Albuquerque then broadcast to other nodes on the network. Concurrently, other nodes actively broadcast the transactions they’ve just created. These broadcasts spread to other nodes on a global network and it takes time to propagate those transactions due to network latency. Regardless of where on the global network a transaction originates, the blockchain protocol places all new transactions in a transaction pool of unverified transactions.

Proof-of-Work and Proof-of-Stake In a blockchain that issues a reward for proof-of-work, miner nodes aggressively select transactions from the transaction pool. It behooves the miner node to verify each transaction while constructing a candidate block because a block containing any bad transactions will be immediately rejected by other nodes—and that would mean the work done by the node was for naught.

Recall from my previous article that each node is in a race to find a nonce for the candidate block it has constructed so it can earn a financial reward and recover energy costs incurred while demonstrating proof-of-work. As of this writing, the current financial reward on the Bitcoin blockchain is 12.5 Bitcoin (BTC), which amounts to roughly \$100,000 USD. Sometimes the financial reward is a transaction fee and sometimes it’s a financial reward plus a transaction fee. What’s important to understand about proof-of-work is that nodes must expend energy and incur equipment and infrastructure costs in order to profitably continue mining blocks; for a node to be sustainable, those costs must be offset by revenue.

It’s no wonder, then, that as soon as a miner finds a nonce it immediately broadcasts that block to every other node on the network in the hope that its just-mined block is added to the end of the blockchain. The Bitcoin blockchain calibrates its nonce difficulty so that new nonces are discovered roughly every 10 minutes, so a lag of even a few seconds can mean that another miner might also find a nonce and broadcast its candidate block.

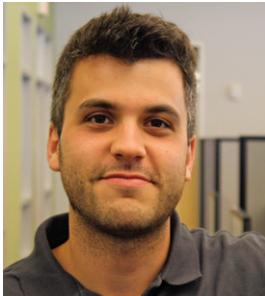
To appreciate the implications of losing the mining race, consider the mining nodes that didn’t find a nonce in time: All the energy they expended was wasted. The miners who didn’t find a nonce have no choice but to stop processing the current block and start all over by grabbing and verifying transactions from the transaction pool. The reason they must stop mining as soon as they learn that another miner found a nonce is that a candidate block has a backlink to the hash of the previous block on the blockchain. When another miner mines a verified block that links to the previous block, the losing miners must create a new block that references the hash for the newly mined block. The losing miners must also discard the transactions they previously selected and choose a new set from the transaction pool because other nodes will reject any new block that contains transactions already included in a previous block.

A node must bear all of the costs required to support its mining equipment. As the Bitcoin blockchain has grown, this has led to another kind of race—a race for the most powerful mining equipment. The more computing power a mining node can access, the more likely it can win each 10-minute race to solve the cryptographic puzzle required to find a nonce.

To appreciate the implications of losing the mining race, consider the mining nodes that didn’t find a nonce in time: All the energy they expended was wasted.

One common criticism of proof-of-work is that it encourages the construction of ever-more-powerful computing centers and the use of increasing amounts of electrical power. A competitive edge is given to the owner of the most powerful computing equipment on proof-of-work-powered blockchain networks. For example, multimillion-dollar datacenters now work exclusively toward mining bitcoin. According to digiconomist.net, Bitcoin’s annual blockchain energy consumption as of June 2018 is 71.12 TWh, which is similar to Chile’s annual energy consumption (bit.ly/2vAdzdl).

Another widely discussed consensus algorithm is proof-of-stake, which rewards nodes that demonstrate an economic stake in the network. Arguably, proof-of-stake’s greatest appeal is that it’s more energy-efficient. Furthermore, it doesn’t issue a cryptocurrency reward for mining a block, although it does issue transaction fees as a reward. It also doesn’t require a race to find the nonce that solves a cryptographic puzzle. Instead, the network randomly selects a node that has registered itself as a “forger” (analogous to Bitcoin’s “miner”) based on the total value and age of its cryptocurrency units. Various implementation details strive to ensure fairness and randomness in selecting among forgers. For example, once a forger is selected it often can’t participate in another round of forging for at least 30 days. Effectively, high-value forger nodes containing the oldest cryptocurrency coins have an edge over other forger nodes.



LEADTOOLS Imaging SDKs Modeling Excellence and Dependability

A Q&A with Hadi Chami, Developer Support Manager

What is new in LEADTOOLS Version 20?

Our biggest goal for Version 20 was to make LEADTOOLS available on every major platform. To accomplish this, we extended our technology with new libraries leveraging .NET Standard targeting .NET Framework, .NET Core, and Xamarin app models.

With these new platforms also came new delivery mechanisms. A majority of our .NET libraries are now available via NuGet packages. NuGets were heavily requested by our customer base and we made sure to provide thoughtfully bundled feature groups developers are already using. Putting LEADTOOLS on NuGet.org helps customers get coding faster and stay up to date with current builds.

LEADTOOLS is a pretty comprehensive toolkit.

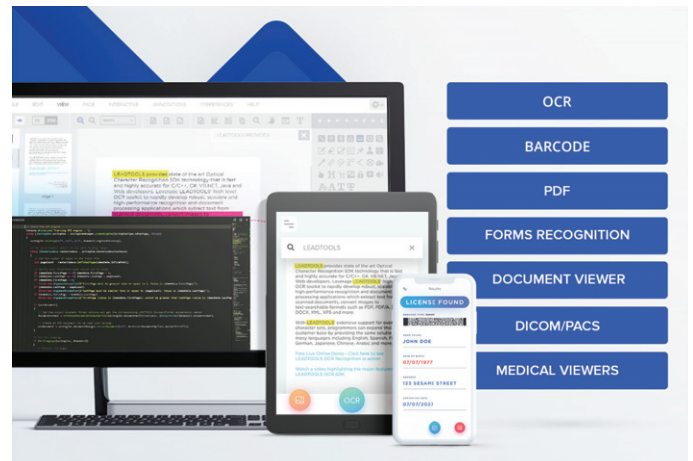
What could you say LEADTOOLS is best known for?

That's a tough question! Like you said, LEADTOOLS is very comprehensive with SDKs for Document, Medical, Multimedia, and Raster Imaging. In recent years, OCR and Document Viewing have been our top performing technologies. We have our own, in-house developed OCR engine we are constantly improving, as opposed to putting a stagnant wrapper around a 3rd party engine. The LEAD engine is very fast and accurate, and our head OCR developer just finished up his PhD. thesis on character recognition.

Our HTML5/JavaScript Document Viewer is also a headlining component for us. The modern workforce relies heavily on digital documents, but it's not just TIFF and PDF anymore. Word, Excel, RTF, Text, PNG, and JPEG files are commonly in the mix whenever you look at someone's cloud storage or ECM. Our Document Viewer can treat all formats uniformly, allowing you to search text, convert, save/edit annotations, etc. Best of all, as a web application that runs on Windows or Linux servers, it's incredibly secure and adaptable, providing scalable zero-footprint document viewing to any desktop, tablet, or smart phone with a web browser.

What sets LEADTOOLS apart from other commercial and open-source SDKs?

Finding the right SDK is tough enough, but many applications need multiple libraries. LEADTOOLS is broad enough for you to standard-



ize on LEADTOOLS for all imaging requirements and eliminate multiple-vendor headaches.

Stack Overflow is great, but sometimes you need a personal touch. Our Support can prove to be a huge advantage over many open-source options where you are largely on your own. LEADTOOLS isn't a side-project developed in our spare time, it's our core product. We are a twenty-eight-year-old company, and are committed to building, maintaining, and supporting our technology.

How is LEADTOOLS adapting to the new ways developers are writing applications?

In addition to offering NuGet packages, LEAD created LEADTOOLS Cloud Services, an alternative development framework. This high-powered, scalable Web API gives developers a hassle-free interface for integrating advanced recognition and document conversion into any application. LEADTOOLS Cloud Services provides all the speed, accuracy, and reliability you have come to expect from LEADTOOLS but is neatly packaged into a simple, pay-as-you-go Web API hosted on Microsoft Azure. With minimal requirements, developers can use their API key to make JSON requests via several programming languages, including PHP, Python, and Perl. Additionally, a wider array of tools and IDEs can be used including Visual Studio Code, Sublime, and Notepad++.

Proof-of-stake supporters make the good point that the cost of running a node is much lower, encouraging more participation and a greater level of decentralization. Ironically, however, proof-of-stake systems discourage the use of the cryptocurrency that blockchain is designed to transact because spending will reduce the node's total value and decrease the chance of being selected as a forger.

One thing to ponder is the point made by blockchain expert Andreas Antonopoulos: "Proof-of-work is also a proof-of-stake but proof-of-stake is not also proof-of-work." He explains that proof-of-work offers a combination of both consensus algorithms by making the point that while miners participating in a proof-of-work-powered network aren't selected based on number or age of cryptocurrency units, miner nodes effectively demonstrate their economic investment in the network by funding the energy required to participate. Thus, the "stake" in the proof-of-work scheme, he argues, is the cost of electricity a node is willing to incur in an effort to successfully mine a block (watch Antonopoulos speaking at a Silicon Valley Bitcoin Meetup on Sept. 13, 2016: bit.ly/2MDfkA1).

Longest Chain The blockchain network constantly extends, branches and prunes itself. The entire view of the blockchain is called the block tree; each miner node actively mines against the block that terminates the block tree's longest chain. You might think that the longest chain is defined by the chain with the greatest number of blocks, but it's actually defined as the sequence of blocks from the genesis block that produces the greatest amount of work. You can derive total work by summing the "difficulty" of each block—a measurement of how unlikely it is to discover a nonce for a candidate block. The network protocol maintains that value, which the Bitcoin blockchain adjusts every 2,016 blocks so that blocks take roughly 10 minutes of processing time to mine. The difficulty value is stored in each block so that work can be computed by nodes that are trying to identify the longest chain.

Occasionally, it's inevitable that two nodes, A and B, will demonstrate proof-of-work by mining a new block within seconds or even milliseconds of each other. Because each node adds its new block to the end of what it sees as the longest chain before broadcasting that block to the network, a fork (branch) in the block tree will appear. Depending on where these nodes are located and the

bandwidth of connected nodes on the network and other latency considerations, some fraction of the network will first see Block A as the new block and will add that to the end of the chain. The other fraction of the network will see Block B as the new block and will add that to the end of the chain. This leaves some nodes with Block A and others with Block B as the terminating block (see **Figure 3**).

When a fork occurs as shown at the top of **Figure 3**, two chains are on the block tree—they're equal in length and both are valid. The problem this presents is clear when you consider that mining nodes look for the longest chain before they begin mining because they need the hash for that chain's terminating block.

If a miner successfully mines Block C and was working on Chain A, it will add Block C to the end of the chain that has Block A as its terminating block (see the bottom block tree in **Figure 3**). Once it does that, it broadcasts Block C to the network and other nodes will see that Chain A is the longest chain. Nodes working on Chain B will see that Chain A is longer than Chain B and will stop mining their current block so they can begin to mine a new block that extends Block C on Chain A. As soon as this happens, the network releases all of the transactions in Block B back into the transaction pool so that they can be picked up in a new round of mining.

You might wonder what happens to the Bitcoin earned by the miner that created Block B: Its transaction commissions and block rewards are never issued. On the Bitcoin network, these rewards aren't given to a miner until 100 blocks have been successfully mined on top of the block in question.

In this article, I explored in greater detail some of the topics introduced in my previous article. Together, the two articles cover most of the fundamental concepts you must truly grasp to understand how blockchains work. After reading both, you should understand blockchain's decentralized, distributed network architecture; SHA-256 hashing; PKC and ECDSA basics; how nodes construct transactions on hash chains and how digital signatures authorize the transfer of digital-asset ownership; how transactions in the transaction pool await selection and verification before getting confirmed into a block; how specialized nodes employ a particular consensus algorithm (such as "miners" using proof-of-work or

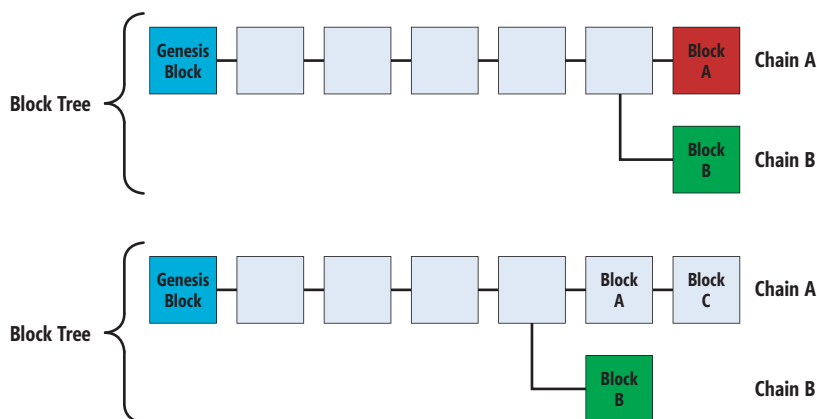


Figure 3 (Top) A Block Tree Showing a Block Tree Fork and Two Equal-Length Chains (Bottom); A Block Tree Showing a Block Tree Fork and One Longest Chain

"forgers" using proof-of-stake) to generate a block; and how nodes on the network add generated blocks to the longest chain. If you want to delve deeper into blockchains, I highly recommend the books and videos available at Safari Books Online (safaribooksnline.com) and those published by Andreas Antonopoulos (antonopoulos.com). ■

JONATHAN WALDMAN is a Microsoft Certified Professional software engineer, a solutions architect with deep technical exposure to a variety of industries and a specialist in software ergonomics. Waldman is a member of the Pluralsight technical team and currently leads institutional and private-sector software-development projects. He can be reached at jonathan.waldman@live.com and followed on Twitter: @jwpulse.

THANKS to the following Microsoft technical expert who reviewed this article: James McCaffrey

Instantly Search Terabytes of Text

Executive Summary

- The dtSearch enterprise and developer product line instantly searches terabytes of text, with no limit on the number of concurrent search threads.
- dtSearch's own document filters support a wide variety of data formats, including "Office" files, PDFs, emails and attachments, online data and other databases.
- The products offer over 25 hit-highlighted search options, with special forensics search features and international language support.
- For developers, the dtSearch Engine offers faceted searching and other advanced data classification options.
- SDKs span a wide variety of online, mobile and other platforms, with APIs for .NET, C++ and Java; new cross-platform .NET Standard/.NET Core/Xamarin SDK.

Key Benefits

Terabyte Indexer. dtSearch enterprise and developer products can index over a terabyte of text in a single index, spanning multiple directories, emails and attachments, online data and other databases. dtSearch products can create and search any number of indexes, and can search indexes during updates.

Concurrent, Multithreaded Searching. dtSearch developer products support efficient multithreaded searching, with no limit on the number of concurrent search threads.

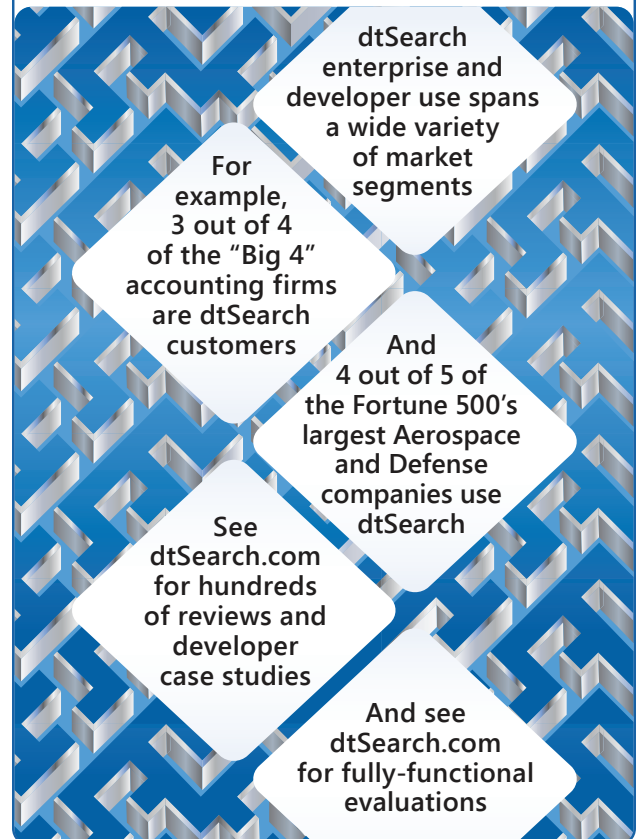
Document Filters and Other Supported Data Types. dtSearch's own document filters cover "Office" documents, PDFs, compression formats, emails and multilevel nested attachments, databases and online data. A DataSource API covers databases like SharePoint, NoSQL and SQL, along with BLOB data.

Over 25 Search Options. The dtSearch product line has over 25 search features, including federated searching with integrated relevancy-ranking options and **multicolor highlighted hits** across online and offline data.



dtSearch®

The Smart Choice for Text Retrieval® since 1991



Forensics; International Languages. dtSearch products offer special forensics search options like the ability to identify credit card numbers and email addresses, and the ability to generate and search for hash values. For international languages, dtSearch products support Unicode, including right-to-left languages, and Chinese/Japanese/Korean character handling options.

Faceted Search and Other Data Classification Options. The dtSearch Engine supports user-interface-driven faceted or "drill down" category searching, as well as numerous other full-text and metadata classification options.

SDKs. SDKs span a wide variety of online, mobile and other platforms, with APIs for .NET, C++ and Java, along with a new cross-platform .NET Standard/.NET Core/Xamarin SDK. Document filters are built into the product line and also available for separate licensing.

For hundreds of developer case studies and press review, and fully-functional evaluations (including the search engine and the document filters), visit



[dtSearch.com](https://www.dtsearch.com)

Visual Studio **LIVE!**

EXPERT SOLUTIONS FOR .NET DEVELOPERS

October 7-11, 2018
Hilton Resort

San Diego

Visual Studio **LIVE!** **25**
YEARS OF CODING INNOVATION

Code With Us in Sunny San Diego!

For the FIRST TIME EVER in our 25 year history, Visual Studio Live! is heading to San Diego, CA for up to 5 days of practical, unbiased, Developer training, including NEW intense hands-on labs.

Join us as we dig into the latest features of Visual Studio 2017, ASP.NET Core, Angular, Xamarin, UWP and more. Code with industry experts, get practical answers to your current challenges, and immerse yourself in the Microsoft platform. Plus, help us celebrate 25 years of coding innovation and experience the education, knowledge-share and networking at #VSLive25.



DEVELOPMENT TOPICS INCLUDE:



**DevOps in the
Spotlight**



**Cloud, Containers
and Microservices**



**AI, Data and
Machine Learning**



**Developing New
Experiences**



**Delivery and
Deployment**



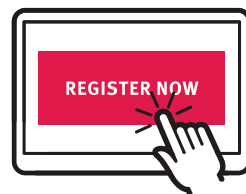
**.NET Core
and More**



**Full Stack
Web Development**



Hands-On Labs



vslive.com/sandiego

Join Us Today!

**Save \$300 When You
Register by August 3**

Use Promo Code MSDN

SUPPORTED BY



PRODUCED BY



DevOps in the Spotlight		Cloud, Containers and Microservices		AI, Data and Machine Learning		Developing New Experiences		Delivery and Deployment		.NET and More		Full Stack Web Development	
START TIME	END TIME	Pre-Conference Full Day Hands-On Labs: Sunday, October 7, 2018 <small>(Separate entry fee required)</small>											
7:00 AM	8:00 AM	Hands-On Lab Registration - Coffee and Morning Pastries											
8:00 AM	5:00 PM	HOL01 Full Day Hands-On Lab: Xamarin and Azure: Build the Mobile Apps of Tomorrow - <i>Laurent Bugnion & Matthew Soucoup</i>						HOL02 Full Day Hands-On Lab: Building a Modern DevOps Pipeline with ASP.NET and Visual Studio Team Services - <i>Brian Randall</i>					
START TIME	END TIME	Pre-Conference Workshops: Monday, October 8, 2018 <small>(Separate entry fee required)</small>											
8:00 AM	9:00 AM	Pre-Conference Workshop Registration - Coffee and Morning Pastries											
9:00 AM	6:00 PM	M01 Workshop: Modern Security Architecture for ASP.NET Core - <i>Brock Allen</i>				M02 Workshop: SQL Server for Developers - <i>Andrew Brust and Leonard Lobel</i>				M03 Workshop: Distributed Cross-Platform Application Architecture - <i>Rockford Lhotka and Jason Bock</i>			
6:45 PM	9:00 PM	Dine-A-Round											
START TIME	END TIME	Day 1: Tuesday, October 9, 2018											
7:00 AM	8:00 AM	Registration - Coffee and Morning Pastries											
8:00 AM	9:15 AM	T01 An Introduction to TypeScript - <i>Jason Bock</i>			T02 Creating Four Beautiful Apps At Once with Xamarin.Forms - <i>Matthew Soucoup</i>			T03 Building Modern Web Apps with Azure - <i>Eric D. Boyd</i>			T04 DevOps on the Microsoft Stack: Adopt, Adapt and Survive - <i>Wouter de Kort</i>		
9:30 AM	10:45 AM	T05 ASP.NET Core 2 for Mere Mortals - <i>Philip Japikse</i>			T06 Building Cross Device Experiences with Project Rome - <i>Tony Champion</i>			T07 Introduction to Azure Cosmos DB - <i>Leonard Lobel</i>			T08 Azure DevOps with VSTS, Docker, and K8 - <i>Brian Randall</i>		
11:00 AM	12:00 PM	KEYNOTE: To Be Announced - <i>Beth Massi, Senior Product Marketing Manager, .NET, Microsoft</i>											
12:00 PM	1:00 PM	Lunch											
1:00 PM	1:30 PM	Dessert Break - Visit Exhibitors											
1:30 PM	2:45 PM	T09 Assembling the Web—A Tour of WebAssembly - <i>Jason Bock</i>			T10 Lessons Learned from Making Resilient Apps with Azure Mobile App Services - <i>Matthew Soucoup</i>			T11 Modern SQL Server Security Features for Developers - <i>Leonard Lobel</i>			T12 Getting to the Core of C# 7 - <i>Adam Tuliper</i>		
3:00 PM	4:15 PM	T13 JavaScript for the C# (and Java) Developer - <i>Philip Japikse</i>			T14 Enhancing UWP Experiences with Fluent Design - <i>Tony Champion</i>			T15 Predicting the Future Using Azure Machine Learning - <i>Eric D. Boyd</i>			T16 Getting to the Core of .NET Core - <i>Adam Tuliper</i>		
4:15 PM	5:30 PM	Welcome Reception											
START TIME	END TIME	Day 2: Wednesday, October 10, 2018											
7:30 AM	8:00 AM	Registration - Coffee and Morning Pastries											
8:00 AM	9:15 AM	W01 Angular 101 - <i>Deborah Kurata</i>			W02 Flying High with Xamarin! - <i>Sam Basu</i>			W03 Developing for Azure Using Visual Studio Code - <i>Brady Gaster</i>			W04 From Continuous Problems to Continuous Deployment - <i>Wouter de Kort</i>		
9:30 AM	10:45 AM	W05 Securing Web APIs from JavaScript/SPA Applications - <i>Brock Allen</i>			W06 Cross-Platform Development with Xamarin, C#, and CSLA .NET - <i>Rockford Lhotka</i>			W07 Docker Containers on Azure—Let Me Count the Ways - <i>Michele Leroux Bustamante</i>			W08 DevOps for the SQL Server Database - <i>Brian Randall</i>		
11:00 AM	12:00 PM	General Session: To Be Announced - <i>Laurent Bugnion, Senior Global Azure Advocate, Microsoft</i>											
12:00 PM	1:00 PM	Birds-of-a-Feather Lunch											
1:00 PM	1:30 PM	Dessert Break - Visit Exhibitors - Exhibitor Raffle @ 1:15pm (Must be present to win)											
1:30 PM	2:45 PM	W09 N Things You Didn't Know About the Router - <i>Deborah Kurata</i>			W10 Essential Tools for Xamarin Developers! - <i>Sam Basu</i>			W11 Building Business Applications Using Bots - <i>Michael Washington</i>			W12 DevSecOps: Securing Applications with DevOps - <i>Wouter de Kort</i>		
3:00 PM	4:15 PM	W13 Getting to the Core of ASP.NET Core Security - <i>Adam Tuliper</i>			W14 Azure, Windows and Xamarin: Using the Cloud to Power Your Cross-platform Applications - <i>Laurent Bugnion</i>			W15 Message-Based Microservices Architectures—Driven by Docker - <i>Michele Leroux Bustamante</i>			W16 Introducing Automated Testing into Legacy Code - <i>David Corbin</i>		
4:30 PM	5:45 PM	W17 Securing Web APIs from Mobile and Native Applications - <i>Brock Allen</i>			W18 Build Awesome AF Apps! - <i>Rachel Appel</i>			W19 Using The Microsoft Cognitive Custom Vision Service - <i>Michael Washington</i>			W20 Is Minimal Really Sufficient? A Look at Work Planning, Execution and Analytics - <i>David Corbin</i>		
7:00 PM	10:30 PM	Out On The Town - Explore the Downtown Gaslamp District!											
START TIME	END TIME	Day 3: Thursday, October 11, 2018											
7:30 AM	8:00 AM	Registration - Coffee and Morning Pastries											
8:00 AM	9:15 AM	TH01 Build Data Driven Web Apps using ASP.NET Core - <i>Rachel Appel</i>			TH02 Modernizing the Enterprise Desktop Application - <i>Oren Novotny</i>			TH03 Coaching Skills for Scrum Masters & The Self-Organizing Team - <i>Benjamin Day</i>			TH04 C# 7, Roslyn and You - <i>Jim Wooley</i>		
9:30 AM	10:45 AM	TH05 Tools for Modern Web Development - <i>Ben Hoelting</i>			TH06 An Architect's Guide to Data Science - <i>Becky Isserman</i>			TH07 Signing Your Code the Easy Way - <i>Oren Novotny</i>			TH08 Sharing C# Code Across Platforms - <i>Rockford Lhotka</i>		
11:00 AM	12:15 PM	TH09 Cryptography 101 - <i>Robert Boedigheimer</i>			TH10 Knockout: R vs Python for Data Science - <i>Becky Isserman</i>			TH11 Unit Testing & Test-Driven Development (TDD) for Mere Mortals - <i>Benjamin Day</i>			TH12 Improving Code Quality with Static Analyzers - <i>Jim Wooley</i>		
12:15 PM	1:15 PM	Lunch											
1:15 PM	2:30 PM	TH13 MVVM and ASP.NET Core Razor Pages - <i>Ben Hoelting</i>			TH14 Analytics and AI with Azure Databricks - <i>Andrew Brust</i>			TH15 How to Interview a Developer - <i>Billy Hollis</i>			TH16 Unit Testing Client-Side Applications - <i>Allen Conway</i>		
2:45 PM	4:00 PM	TH17 HTTP/2: What You Need to Know - <i>Robert Boedigheimer</i>			TH18 Power BI: What Have You Done for Me Lately? - <i>Andrew Brust</i>			TH19 The Most Important Lessons I've Learned in Forty Years of Developing Software - <i>Billy Hollis</i>			TH20 Migrating from AngularJS to Angular + TypeScript - <i>Allen Conway</i>		

Speakers and sessions subject to change

CONNECT WITH US



twitter.com/vslive – @VSLive



facebook.com – Search "VSLive"



linkedin.com – Join the "Visual Studio Live" group!

vslive.com/sandiego

Mixed Reality and Fluent Design

Tim Kulp

Microsoft has been pushing hard at mixed reality (MR) development since it introduced HoloLens in 2014, and at the Build 2018 conference in May, it provided a wealth of great insight into new features in the Unity development environment for MR, as well as UI design best practices. In this article, I'll leverage the technical guidance presented at Build to show how the Microsoft Fluent Design System can be used to build immersive experiences in MR—specifically, a HoloLens app that allows non-verbal children to communicate via pictograms.

Microsoft's Fluent Design system uses three core principles to deliver amazing experiences. They are:

Adaptive: MR must bridge and mingle the real world and digital components to create a holistic experience. As such, the UI must be mindful of the environment, while enhancing (but not replacing) the user's real-world experience.

This article discusses:

- How to use Fluent Design in mixed reality for adaptive, empathetic and beautiful designs
- How to implement layout controls from the Mixed Reality Toolkit to organize your mixed reality UI
- Using the Receiver model to centralize reactions to Interactables
- Implementing the Text to Speech component from the Mixed Reality Toolkit

Technologies discussed:

C#, Unity 2018, HoloToolkit, Mixed Reality Design Lab, HoloLens

Empathetic: Empathetic design in MR focuses on understanding the user's intent and needs within the app experience. As an example, not all users can use their hands for gestures, so the HoloLens clicker and voice commands provide input modalities that are empathetic to the user's needs.

Beautiful: Beautiful apps are a major challenge in MR. As a designer or developer, you must extend the real-world experience without overwhelming it. Careful planning and the right tools are needed.

Unity is the most common tool for building MR experiences. According to Unity, 90 percent of HoloLens and MR apps are built with its development environment. While Unity has its own UI system, it doesn't easily translate to Fluent Design. The team at Microsoft has built the Mixed Reality Toolkit (bit.ly/2yKw2r) and Mixed Reality Design Lab (bit.ly/2MskePH) tools, which help developers build excellent UXes through the configuration of components. Using these projects, developers can avoid complex graphic design work and focus on the functionality of their app. Both the Mixed Reality Toolkit and Design Lab are well documented with many code examples. In this article I use the Mixed Reality Toolkit 2017.4.0.0 Unity Package to build an interface and show you how to pull the various components together into your own cohesive, unique interface.

Before you set up your Unity environment, make sure to complete the Getting Started tutorial from the Mixed Reality Toolkit (bit.ly/2KcVvIN). For this project, I'm using the latest (as of this writing) Unity build 2018.2.08b. Using Unity 2018.2 required an update to the Mixed Reality Toolkit, but the upgrade did not cause any conflicts for the functionality implemented in this article.

Transform the way modern business is built and run

-  Application Performance
-  End User Monitoring
-  Infrastructure Visibility
-  Business Performance
-  Cloud Migration



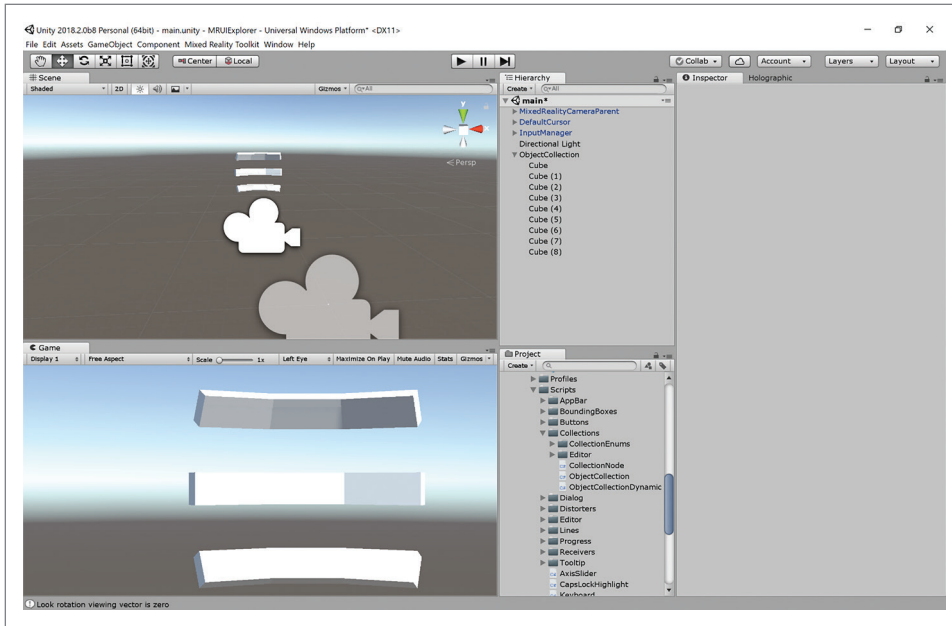


Figure 1 Updating the Surface Type to Sphere

Designing Your UI for MR

An excellent MR experience starts with an effective UI flow. Unlike Web or mobile apps, the UI of an MR app has one big unknown: the environment. In Web development you must be mindful of the screen resolution and browser capabilities, while in mobile development its pixel density and phone capabilities. But in MR the variable nature of the user's physical surroundings and environment becomes paramount.

By targeting a specific environment I can tune for the physical space and for the specific capabilities of the device being used. In my example app, I'm designing for a classroom setting where students will use the app with their teacher. This is a well-lit and relatively quiet environment, which is very different from a warehouse or night club interior design. Plan your UI based on the environment.

The environment is not only external to the user but also driven by the user's device capabilities. When thinking about my UI, I need to be aware of the possible controls to use. Thinking of my user's environment and capabilities, speech commands will not play a key role due to lack of verbal communication. Gestures would be limited as my target users could also have fine motor skill challenges. This leads to large buttons and the users having tools like the HoloLens clicker to select the buttons.

With that in mind, I need to select how to layout my UI for a beautiful design. If you're a Web developer with Bootstrap (or similar UI framework), you're familiar with building to the Grid. In UWP, you have layout controls like StackPanel, Grid and Canvas. In the Mixed Reality Toolkit you have the Object Collection Component.

The Object Collection Component can be found in the Mixed Reality Toolkit under Assets | UX | Scripts | Object Collection. To use it, create an Empty game object in your scene. Name the empty game object "MainContainer" and then add the Object Collection component to the empty game object. For testing purposes,

add a cube as child object to the MainContainer. Set the cube's scale to be 0.5x, 0.2y, 0.1z so it looks like a thin rectangle. Now duplicate that cube eight times to have a total of nine cubes as child objects to the MainContainer.

Once the cubes are added, press the Update Collection button in the Inspector to apply the Object Collection layout to the child objects. By default, the object collection uses the Surface Type of Plain. For my UI, I want the collection to wrap around the user to feel more immersive. To do this, update the Surface Type for the object collection to Sphere in the Inspector. Now all the cubes should appear as shown in **Figure 1**. Object collections provide multiple Surface Types to allow your UI to engage

the user for your specific scenario. Later in the article I'll show you how to use a different Surface Type to achieve a different goal.

What Are We Building?

Non-verbal children have books of pictograms that they use with a teacher to express their needs. Each pictogram is a word or phrase that the child and teacher build together. In this article, I'll show you how to build an MR experience to take the book into the HoloLens, so students can build a sentence to communicate with their teachers.

In UWP, you have layout controls like StackPanel, Grid and Canvas. In the Mixed Reality Toolkit you have the Object Collection Component.

I'll start by using the Object Collection to add a bunch of buttons for the user to select to build a sentence. The buttons will be the pictograms. Start by adding a Resources | Data folder into your project window. Inside the data folder create a words.json file to represent the words in the app. The sample code employs a short words file to keep things simple.

In the project window, create a Scripts folder and inside that folder create a new C# script called WordLoader. This component reads the words.json file and converts it into a collection of C# Word objects. In the Design Lab there are example projects that illustrate the finished sample of reading data from a file and displaying it in an MR experience. I'm adapting the code from the

Manipulating Documents?

APIs to view, convert, annotate, compare, sign, assemble and search documents in your applications.

Try GroupDocs APIs for FREE

Download a Free Trial at

<https://downloads.groupdocs.com>



GroupDocs.Viewer

View over 50 documents and image formats in any application using document viewer APIs.



GroupDocs.Annotation

Add annotations to specific words, phrases and any region of the document.



GroupDocs.Conversion

Fast batch document conversion APIs for any .NET, Java or Cloud app.



GroupDocs.Comparison

Compare two documents and get a difference summary report.



GroupDocs.Signature

Digitally sign Microsoft Word, Excel, PowerPoint and PDF documents.



GroupDocs.Assembly

Document automation APIs to create reports from templates and various data sources.



GroupDocs.Metadata

Organize documents with metadata within any cross platform application.



GroupDocs.Search

Transform your document search process for advance full text search capability.

► GroupDocs.Text

► GroupDocs.Editor

► GroupDocs.Parser

► GroupDocs.Watermark

Americas: +1 903 306 1676

EMEA: +44 141 628 8900
sales@asposeptyltd.com

Oceania: +61 2 8006 6987

Periodic Table project to keep this example concise and familiar, as shown in **Figure 2**. Please check out the Period Table project at bit.ly/2KmSizg for other features and capabilities built on top of reading a data file and binding the results to an object collection.

Next, write the code shown in **Figure 3** for the WordLoader component. This component loads the words from the JSON file and adds them as buttons to the Object Collection. There are two public variables in the component: Parent is the Object Collection that holds the words and WordPrefab is the prefab used to represent the words in the MR experience.

Once all the words are loaded to the Parent, calling Parent.UpdateCollection will arrange the newly created buttons based on the layout of the Object Collection component. Each time you change a property in the Object Collection component, call UpdateCollection to ensure that all game objects are updated properly.

Now in the Unity Editor you create an empty game object called Managers, which holds all the utility components for the scene. Add the WordLoader component to Managers, then set Parent to MainContainer. Finally, set WordPrefab to use the Holographic Button prefab (found in HoloToolkit | UX | Prefabs | Buttons). These buttons implement Fluent Design concepts like using light to reveal focus and action. When the user sees the button light up, he knows what's selected.

Figure 2 The WordLoader Script

```
[System.Serializable]
class Word
{
    public string category;
    public string text;
    public string image;
}

[System.Serializable]
class WordsData
{
    public Word[] words;
    public static WordsData FromJSON(string data)
    {
        return JsonUtility.FromJson<WordsData>(data);
    }
}
```

Figure 3 Loading Words from the JSON File

```
public class WordLoader : MonoBehaviour
{
    public ObjectCollection Parent;
    public GameObject WordPrefab;

    private void OnEnable() {
        if(Parent.transform.childCount > 0)
            return;

        TextAsset dataAsset = Resources.Load<TextAsset>("Data/words");
        WordsData wordData = WordsData.FromJSON(dataAsset.text);

        foreach (Word w in wordData.words) {
            GameObject newWord = Instantiate<GameObject>(WordPrefab, Parent.transform);
            newWord.GetComponent<CompoundButtonText>().Text = w.text;
            newWord.GetComponent<CompoundButtonIcon>().OverrideIcon = true;
            string iconPath = string.Format("Icons/{0}", w.image);
            newWord.GetComponent<CompoundButtonIcon>().iconOverride =
                (Texture2D)Resources.Load<Texture2D>(iconPath);
        }
        Parent.UpdateCollection();
    }
}
```

Running the app now creates buttons for each word in the word data file (see **Figure 4**). The Holographic buttons will surround the user in a sphere. While in Play Mode, feel free to update the Object Collection's Surface Type to experiment with other layouts. Adjust the cell spacing and row counts to find the right mix for your experience. Remember to Update Collection after each change to see the change in the editor.

Keeping up with the User

Once the Object Collection is built it's set to specific coordinates in the world. As the user moves, the Object Collection will stay where it was originally created. For a classroom this doesn't work because children don't always stay at their desk. I need to update the UI to stick with the user as they move around the real world. In Fluent Design you need to be adaptive to the environment, even when that person moves and changes the surroundings.

At the Build 2018 conference, a technical session ("Building Applications at Warehouse Scale for HoloLens, bit.ly/2yNmwxT) illustrated the challenges with building apps for spaces larger than a table top. These range from keeping UI components visible when blocked by objects like a forklift in a warehouse to keeping the UI accessible as the user moves around a space for projects like interior design. The Mixed Reality Toolkit gives you some tools to solve some of the issues by using the Solver System.

The Solver System allows game objects to adjust their size or position based on the user's movement through the world. In Unity, you can see how some Solvers work in the scene using the editor. To add a Solver, select the MainContainer object and add the SolverBodyLock component (found in HoloToolkit | Utilities | Scripts | Solvers). The SolverBodyLock component allows the sphere of buttons to stay with the user, because as the user moves throughout the space the MainContainer moves with them. To test this, run your app in the editor and move around the world space using the arrow or WASD keys. You'll notice forward, backward and side-to-side movements keep the MainContainer with you. If you rotate the camera, the MainContainer doesn't follow you.

To make the UI follow the user as he rotates in space, use the SolverRadialView, which by default keeps the object just inside the user's peripheral view. How much of the object is visible can increase or decrease based on the component's settings. In this use case, I don't want the UI chasing the user as he turns away, so the SolverBodyLock is enough. It keeps the list of buttons with the user, but not always in their face.

Maintaining Size

In my app I expect people to move through a classroom setting. In MR, just like the real world, when people move away from an object, that object appears smaller. This creates problems: Using Gaze to target UI components that are on distant objects can be difficult, and icons can be difficult to discern from far away. The ConstantViewSize Solver addresses this by scaling the UI component up or down based on the user's distance.

For this sample app you're going to add a component that lets users build sentences. Each button pressed in the MainContainer adds to the sentence. For example, if the user wants to say, "I am

**IS BAD DATA THREATENING
YOUR BUSINESS?**

CALL IN THE **FABULOUS 4**

IT'S CLOBBERIN' TIME...WITH DATA VERIFY TOOLS!

ADDRESS!

PHONE!

EMAIL!

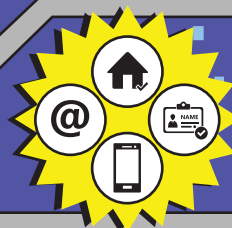
NAME!

Visit Melissa Developer Portal to quickly combine our APIs (address, phone, email and name verification) and enhance ecommerce and mobile apps to prevent bad data from entering your systems.

With our toolsets, you'll be a data hero – preventing fraud, reducing costs, improving data for analytics, and increasing business efficiency.

- Single Record & Batch Processing
- Scalable Pricing
- Flexible, Easy to Integrate Web APIs: REST, JSON & XML
- Other APIs available: Identity, IP, Property & Business

LET'S TEAM UP TO FIGHT BAD DATA TODAY!



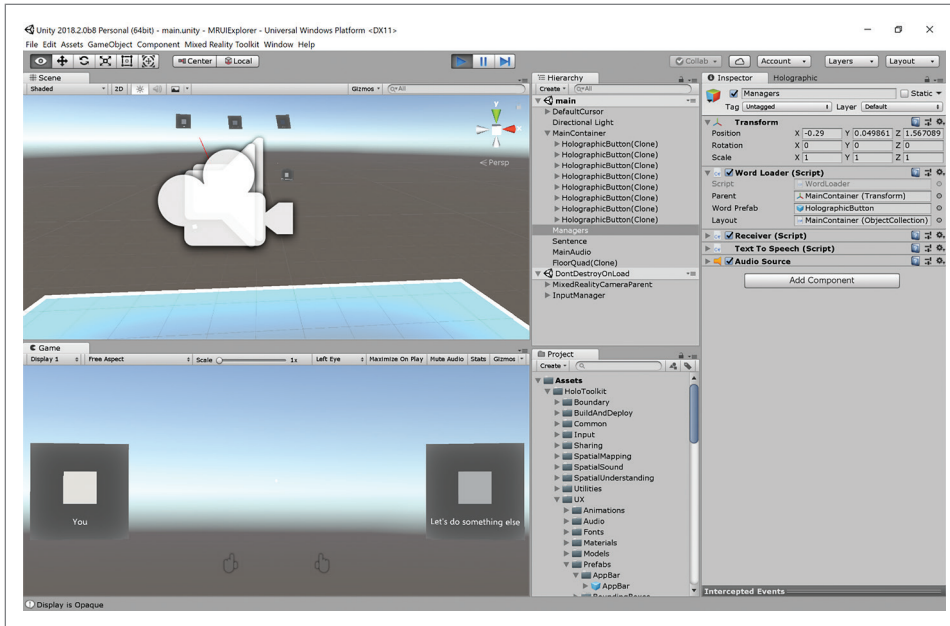


Figure 4 Viewing the Words as Holographic Buttons in the Object Collection

hungry,” he would click the “I” button, then “am” button, then “hungry” button. Each click adds a word to the sentence container.

Create a new empty game object in the Hierarchy and call it Sentence. Add the Object Collection component to Sentence, set the Surface Type to Plane and Rows to 1. This creates a flat interface that won’t break into rows for content. For now, add a cube to the Sentence game object, so that you can see the constant size Solver in action. Later I’ll add the words from the buttons. Set the cube’s scale to $x = 1.4$, $y = 0.15$, $z = 0.1$. If you start the app now and move through the space, the white cube will shrink and grow as you move farther away and closer to it.

To lock the cuboid’s visual size, add the SolverConstantViewSize component to the Sentence object. The scale for the component can also be constrained using the minimum and maximum scale range settings, so you can control how large you want the UI component to grow or shrink and at what distances—this lets you scale components from a table top, to a classroom, to a warehouse.

There are more solvers available in the Mixed Reality Toolkit, including Momentumizer and Surface Magnetism. I encourage you to check out the Solver System and grow your own MR experience with a UI that can work with the user in the real-world space (bit.ly/2bxc4lc).

The Sentence and the Receiver

To enable the buttons with the MainContainer to connect with the Sentence object, I implement a Receiver that receives events from the buttons pressed in the MainContainer and adds the content of that button to the Sentence. Using the Interactable Object and Receiver pattern built into the Mixed Reality Toolkit, connecting the prefab buttons found in the HoloToolkit | UX | Buttons folder is simple.

Create a new C# script called Receiver in the Scripts folder of your project, as shown in Figure 5. Change the script to inherit from MonoBehaviour to InteractionReceiver. Create a public variable to hold a reference to the Sentence game object and then implement

the override method for InputDown to respond to a user’s click action on the Holographic buttons.

Inside of the InputDown method, create a switch statement that checks the name property of the obj game object. For now, create the default condition, which occurs no matter what the game object name is. Within the default statement, I take the calling object and create a new instance of it under the Sentence game object. This creates a new instance of the button in the Sentence object collection without registering it with the Receiver. If you press the button in the Sentence object, nothing happens. Before testing, remove the white cuboid placeholder so you can see the buttons appearing on the Object Collection plane.

To connect the buttons in the MainContainer to the Receiver, you need a few lines of code added to the WordLoader. As the app loads words, each will be registered as an Interactable for the Receiver. Interactables are the list of game objects using the InteractableObject component that the Receiver reacts to when an event occurs. Within the WordLoader onEnable method, add the following code directly under the childCount check:

```
var receiver = GetComponent<Receiver>();
if (receiver == null)
    return;
```

This code checks to ensure that a receiver is present. Then in the foreach loop for the words, add the following code after loading the icon for the button:

```
receiver.RegisterInteractable(newWord);
```

RegisterInteractable adds the game object to the list of interactables for the Receiver. Now clicking on the Holographic buttons in the MainContainer triggers the Receiver to perform the InputDown method and creates a copy of the button that was clicked.

With some minor reconfiguring of the MainContainer Object Collection cell size and placement, you’ll see something like the image shown in Figure 6.

Figure 5 The Receiver C# Script

```
public class Receiver : InteractionReceiver
{
    public ObjectCollection Sentence;

    protected override void InputDown(GameObject obj, InputEventData eventData)
    {
        switch (obj.name)
        {
            default:
                GameObject newObj = Instantiate(obj, Sentence.gameObject.transform);
                newObj.name = "Say" + obj.name;
                Sentence.UpdateCollection();
                break;
        }
    }
}
```




Integrate Documents and Reporting in Any Platform

A Reporting Q&A with Bjoern Meyer, Text Control

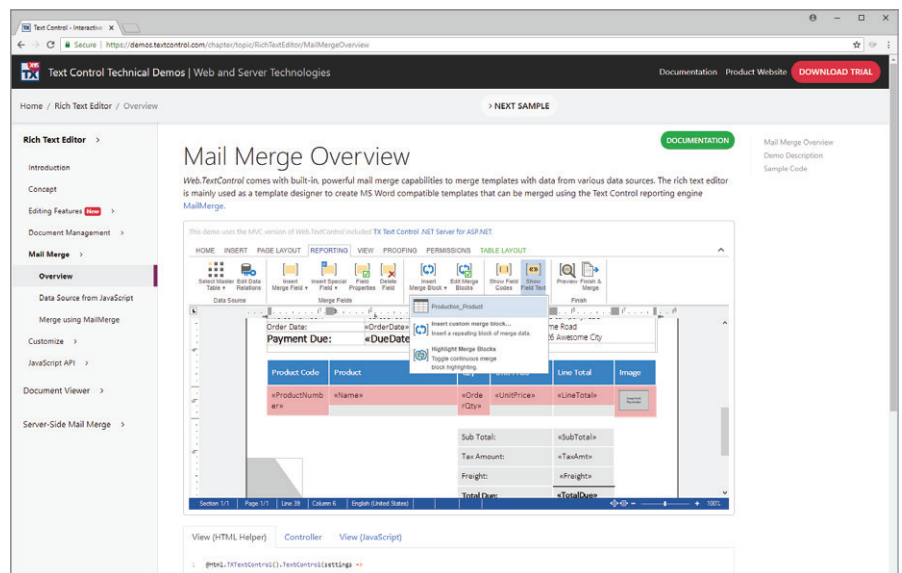
Founded in 1991, Text Control is an award-winning Visual Studio Industry Partner and leading vendor of word processing and reporting components for Windows, web and mobile development technologies.

Q What is Text Control doing?

A Our mission is to use everyday innovation to uncover our user's real reporting requirements. Digital transformation changed every process in today's business world. The number of e-commerce transactions skyrocketed and supply chains are fully connected. In nearly any business process, documents and reports need to be designed, created, shared and archived. Our technologies help companies to integrate document processing to client, web and cloud solutions to gain the largest competitive advantage. We have been developing software components for reporting and document processing for more than 25 years. We are continually looking for new and innovative ways to improve document processing to make these processes easier for end-users and more efficient.

Q What is the Text Control Reporting Framework?

A The Text Control Reporting Framework combines powerful reporting features with an easy-to-use, MS Word compatible word processor. Users can create documents and templates using ordinary Microsoft Word skills. It is completely independent from MS Word or any other third-party application and can be completely integrated into business applications. The Text Control Reporting Framework is included in all .NET based TX Text Control products including ASP.NET, Windows Forms and WPF.



Q What sets Text Control Reporting apart from other reporting vendors?

A Text Control Reporting is based on the powerful word processing component TX Text Control. The MS Word compatible template can be merged with a data object (business object) or database content with one line of code. At the same time, Text Control provides a powerful API to customize this merge process completely. The report generation can be fully integrated into .NET applications.

Q Tell us more about your Cloud reporting Web API

A Text Control ReportingCloud brings complete reporting functionality to the cloud so all developers can use it, irrespective of the platform or language they're using. Its highly RESTful API can be used to merge Microsoft Word compatible templates with JSON data from all clients including .NET, JavaScript, PHP, NodeJS, jQuery, Ruby, Python, Android, Java and iOS.

For more information, visit →

www.textcontrol.com

While I'm using the Holographic buttons in this example, the interactable component can be applied to anything from a custom mesh to text. Let your creativity help you imagine amazing interfaces. Just remember that one of the goals of MR is to merge the digital and physical worlds so that they work together.

Leveraging Text to Speech

Part of Fluent Design is being empathetic to your users' needs. In this example, you're enabling those lacking verbal speech to express themselves. Now you need to use Text to Speech to say out loud what the user has built in her sentence. The HoloToolkit provides an excellent component to do this with minimal code or configuration. If you're not using a HoloLens, check out my *MSDN Magazine* article, "Using Cognitive Services and Mixed Reality" (msdn.com/magazine/mt814418), to see how to use Text to Speech through Azure Cognitive Services.

To start using Text to Speech, add an Audio Source to your project Hierarchy. This is used by the Text to Speech component to generate audio for users to hear. Now add the Text to Speech component (found in HoloToolkit | Utilities | TextToSpeech) to the Managers game object. I use the Managers game object as the home for all utility methods (Receiver, Text to Speech) to centralize events and what happens with those events. In this case the Receiver will use the Text to Speech component to say the sentence using the same Receiver setup you have for adding words to the sentence. Set up the Text to Speech component to use the Audio Source you just created by selecting it in the Inspector.

In the Receiver, I'm going to detect if any words exist in the Sentence, and if so, I'll add a button that will say the Sentence. Add a new game object public variable called SayButton to the Receiver component.

```
public GameObject SayButton;
```

This will be the prefab used as the Say button. Within the switch statement's default block add the following if statement prior to adding the new object to the Sentence:

```
if (Sentence.transform.childCount == 0)
{
    GameObject newSayButton = Instantiate(SayButton, Sentence.transform);
```

```
newSayButton.name = "Say";
newSayButton.GetComponent<CompoundButtonText>().Text = "Say: ";
GetComponent<Receiver>().RegisterInteractable(newSayButton);
}
```

In this case when there aren't any children in the Sentence object, add the Say button prior to any other buttons. This will keep Say to the far left of the Sentence object for any sentence. Next, create a new case statement in the switch to detect the Say button by name. Remember, the switch is based on obj.name and when you created this button you set the name to "Say." Here's the code:

```
case "Say":
    StringBuilder sentenceText = new StringBuilder();
    foreach (CompoundButtonText text in
        Sentence.GetComponentsInChildren<CompoundButtonText>())
    {
        sentenceText.Append(text.Text + " ");
    }
    TextToSpeech tts = GetComponent<TextToSpeech>();
    tts.StartSpeaking(sentenceText.ToString());
    break;
```

This case detects the Say button by name and then loops through all the children of Sentence to build a string of words for the TextToSpeech component to say as a sentence. When the user is done building a statement, he can press the Say button to express his statement. For simplicity's sake I did not have you implement the Text to Speech Manager of the HoloToolkit (bit.ly/2lxjpoq), but it's helpful to avoid situations like a user pressing Say over and over again, resulting in the Text to Speech component trying to say multiple things at the same time.

Wrapping Up

Using Fluent Design with MR is a natural fit. Tools like the Mixed Reality Toolkit and Mixed Reality Design Labs are excellent springboards to build an experience that's adaptive, empathetic and beautiful. In this article I covered how to implement some of the experiences shown at Build 2018 for MR. By using Object Collections as a layout tool, Solver Systems to keep the UI with the user as they move through space, and then connecting the experience with a Receiver for centralized event management, I showed you how to quickly build an MR app.

Using the examples from the Mixed Reality Toolkit, you can take this sample app further with new interface options and capabilities. Review the MR sessions from Build 2018 to get further inspiration on how to build exceptional experiences that blend the physical and digital worlds. ■

TIM KULP is the director of Emerging Technology at Mind Over Machines in Baltimore, Md. He's a mixed reality, artificial intelligence and cloud app developer, as well as author, painter, dad, and "wannabe mad scientist maker." Find him on Twitter: @tim_kulp or via LinkedIn: linkedin.com/in/timkulp.

THANKS to the following technical expert for reviewing this article: Will Gee (Balti Virtual)

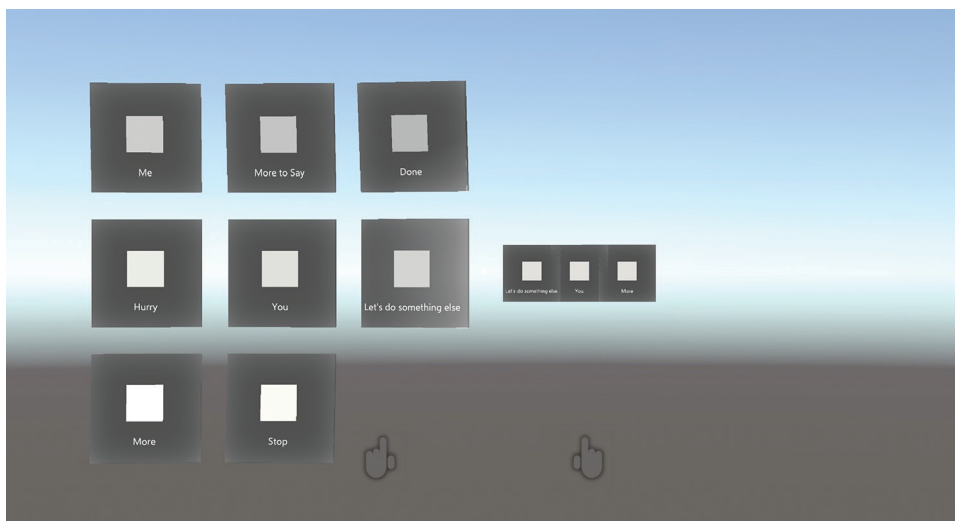


Figure 6 MainContainer and Sentence Object Collections After Update

DocuVieware

Universal HTML5 and Document Management Kit



Easy
integration



Full support for custom
snap-in



Zero-footprint
solution



Fully customizable
UI



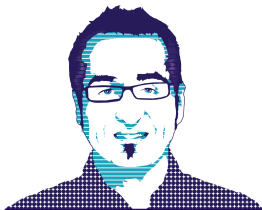
Mobile devices
optimization



Fast & crystal-clear
rendering



Check the New Features and the Online Demos
60-day Free Trial Support Included at www.docuvieware.com



Social-Style Notifications with ASP.NET Core SignalR

Social networks and mobile OSes made pop-up, balloon-style notifications incredibly popular and mainstream, but Microsoft Windows 2000 was probably the first software to extensively use them. Balloon notifications make it possible to communicate notable events to the user without requiring immediate action and attention—unlike conventional pop-up windows. Key to these balloon-style notifications is the underlying infrastructure that delivers the message in real time, right to the open window where the user is working.

In this article, you'll see how to use ASP.NET Core SignalR to produce pop-up notifications. The article presents a sample application that keeps track of logged users and gives each a chance to build and maintain a network of friends. Like in a social-network scenario, any logged user may be added or removed from a friend list at any time. When this happens in the sample application, the logged user receives a context-sensitive notification.

Authenticating Application Users

Balloon-style notifications are not plain broadcast notifications sent to whoever is listening to SignalR messages on a Web socket channel. Rather, they're sent to specific users, logged into the application. Opening and listening to a plain Web socket channel is a good way to approach the problem, but ASP.NET Core SignalR just provides a more abstract programming interface and offers support for alternate network protocols beyond WebSockets.

The first step in the building of the application is adding a layer for user authentication. The user is presented a canonical login form and provides her credentials. Once properly recognized as a valid user of the system, she receives an authentication cookie packed with a number of claims, as shown here:

```
var claims = new[]
{
    new Claim(ClaimTypes.Name, input.UserName),
    new Claim(ClaimTypes.Role, actualRole)
};
var identity = new ClaimsIdentity(claims,
    CookieAuthenticationDefaults.AuthenticationScheme);
await HttpContext.SignInAsync(
    CookieAuthenticationDefaults.AuthenticationScheme,
    new ClaimsPrincipal(identity));
```

The sample code shows how to create an ad hoc *IPrincipal* object in ASP.NET Core around a user name once credentials have been successfully verified. It's key to note that for authentication to work properly in ASP.NET Core, when used in combination with the

newest SignalR library, you should enable authentication in the *Configure* method of the startup class quite early (and in any case, earlier than completing the SignalR route initialization). I'll return on this point in a moment; meanwhile, let's review the infrastructure of friendship in the sample application.

Defining Friends in the Application

For the purpose of the sample application, a friend relationship is merely a link between two users of the system. Note that the demo application doesn't use any database to persist users and relationships. A few users and friend relationships are hardcoded and reset when the application restarts or the current view is reloaded. Here's the class that represents a friend relationship:

```
public class Friendship
{
    public Friendship(string friend1, string friend2)
    {
        UserName1 = friend1;
        UserName2 = friend2;
    }
    public string UserName1 { get; set; }
    public string UserName2 { get; set; }
}
```

As the user logs in, he's served an index page that provides the list of friends. Through the UI of the page, the user can both add new friends and remove existing ones (see **Figure 1**).

When the user types the name of a new friend, an HTML form posts and a new friend relationship is created in memory. If the

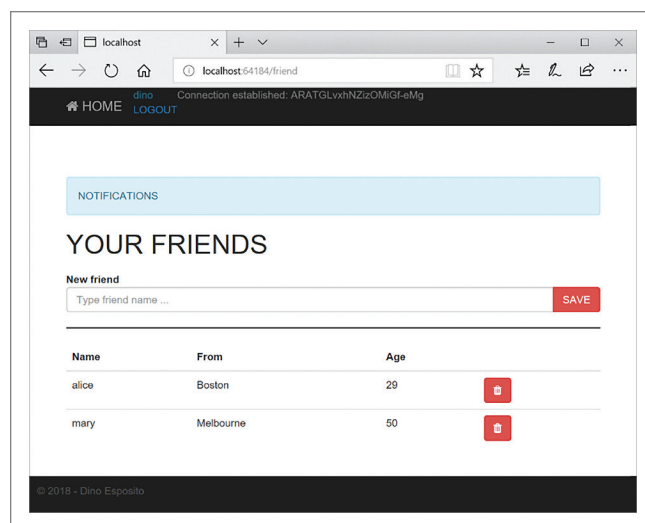


Figure 1 The Home Page of a Logged User

Code download available at bit.ly/2HVYlp5.

typed name doesn't match an existing user, a new User object is created and added to the in-memory list, like so:

```
[HttpPost]
public IActionResult Add(string friend)
{
    var currentUser = User.Identity.Name;
    UserRepository.AddFriend(currentUser, friend);

    // More code here
    ...

    return new EmptyResult();
}
```

As you may notice, the controller method returns an empty action result. It's assumed, in fact, that the HTML form posts its content via JavaScript. Therefore, a JavaScript click handler is attached to the submit button of the form programmatically, as shown here:

```
<form class="form-horizontal" id="add-form" method="post"
      action="@Url.Action("add", "friend")">

    <!-- More input controls here -->

    <!-- SUBMIT button -->
    <button id="add-form-submit-button"
            class="btn btn-danger" type="button">
        SAVE
    </button>
</form>
```

The JavaScript posting code triggers the server-side operation and returns. The new list of friends, whether as a JSON array or an HTML string, can be returned by the controller method and integrated in the current page document object model by the same JavaScript caller code. It works well, but there's a glitch to take into account that could possibly be an issue in some scenarios.

Imagine the user holds multiple connections to the same server page. As an example, the user has multiple browser windows opened on the same page and interacts with one of those pages. In a situation in which the call brings back a direct response (whether JSON or HTML), only the page from where the request originated ends up being updated. Any other open browser window remains static and unaffected. To work around the problem, you can leverage a feature of ASP.NET Core SignalR that lets you broadcast changes to all connections related to the same user account.

Broadcasting to User Connections

The ASP.NET MVC controller class that receives calls to add or remove friends incorporates a reference to the SignalR hub context. This code shows how that's done:

```
[Authorize]
public class FriendController : Controller
{
    private readonly IHubContext<FriendHub> _friendHubContext;

    public FriendController(IHubContext<FriendHub> friendHubContext)
    {
        _friendHubContext = friendHubContext;
    }

    // Methods here
    ...
}
```

As usual in SignalR programming, the friend hub is defined in the startup class, as shown in **Figure 2**.

It's key that the UseSignalR call follows the UseAuthentication call. This ensures when a SignalR connection is established on the given route that information about the logged user

and claims is available. **Figure 3** offers a deeper look at the code that handles the form post when the user adds a new friend to the list.

The Clients property of the hub context has a property called User, which takes a user ID. When invoked on the User object, the SendAsync method notifies the given message to all currently connected browser windows under the same user name. In other words, SignalR has the ability to group automatically all connections from the same authenticated user to a single pool. In light of this, SendAsync invoked from User has the power to broadcast the message to all windows related to the user, whether they come from multiple desktop browsers, in-app Web views, mobile browsers, desktop clients or whatever else. Here's the code:

```
_friendHubContext.Clients.User(currentUser).SendAsync("refreshUI");
```

Sending the refreshUI message to all connections under the same user name ensures that all opened windows are in a way synchronized to the add-friend function. In the sample source code, you'll see that the same thing happens when the currently logged-in user removes a friend from his list.

Configuring the User Client Proxy

In SignalR, the User object has nothing to do with the User object associated with the HTTP context. In spite of the same property name, the SignalR User object is a client proxy and a container of claims. Yet, a subtle relationship exists between the SignalR user-specific client proxy and the object representing the authenticated user. As you may have noticed, the User client proxy requires a string parameter. In the code in **Figure 3**, the string parameter is

Figure 2 Defining The Hub

```
public void Configure(IApplicationBuilder app)
{
    // Enable security
    app.UseAuthentication();

    // Add MVC
    app.UseStaticFiles();
    app.UseMvcWithDefaultRoute();

    // SignalR (must go AFTER authentication)
    app.UseSignalR(routes =>
    {
        routes.MapHub<FriendHub>("/friendDemo");
    });
}
```

Figure 3 Handling the Form Post

```
[HttpPost]
public IActionResult Add(string friend)
{
    var currentUser = User.Identity.Name;
    UserRepository.AddFriend(currentUser, friend);

    // Broadcast changes to all user-related windows
    _friendHubContext.Clients.User(currentUser).SendAsync("refreshUI");

    // More code here
    ...

    return new EmptyResult();
}
```

Figure 4 The Final Add Method Code

```
public IActionResult Add(string addedFriend)
{
    var currentUser = User.Identity.Name;

    // Save changes to the backend
    UserRepository.AddFriend(currentUser, addedFriend);

    // Refresh the calling page to reflect changes
    _friendHubContext.Clients.User(currentUser).SendAsync("refreshUI");

    // Notify the added user (if connected)
    _friendHubContext.Clients.User(addedFriend).SendAsync("added", currentUser);

    return new EmptyResult();
}
```

the name of the currently logged-in user. That's just a string identifier, though, and can be anything you configure it to be.

By default, the user ID recognized by the user client proxy is the value of the NameIdentifier claim. If the list of claims of the authenticated user doesn't include NameIdentifier, there's no chance the broadcast would work. So you have two options: One is to add the NameIdentifier claim when creating the authentication cookie, and the other is to write your own SignalR user ID provider. To add the NameIdentifier claim, you need the following code in the login process:

```
var claims = new[]
{
    new Claim(ClaimTypes.Name, input.UserName),
    new Claim(ClaimTypes.NameIdentifier, input.UserName),
    new Claim(ClaimTypes.Role, actualRole),
};
```

The value assigned to the NameIdentifier claim doesn't matter as long as it's unique to each user. Internally, SignalR uses an IUserIdProvider component to match a user ID to the connection groups rooted to the currently logged user, like so:

```
public interface IUserIdProvider
{
    string GetUserId(HubConnectionContext connection);
}
```

The IUserIdProvider interface has a default implementation in the DI infrastructure. The class is DefaultUserIdProvider and is coded as follows:

```
public class DefaultUserIdProvider : IUserIdProvider
{
    public string GetUserId(HubConnectionContext connection)
    {
        var first = connection.User.FindFirst(ClaimTypes.NameIdentifier);
        if (first == null)
            return null;
        return first.Value;
    }
}
```

As you can see, the DefaultUserIdProvider class uses the value of the NameIdentifier claim to group user-specific connection IDs. The Name claim is meant to indicate the name of the user, but not necessarily to provide the unique identifier through which a user is identified within the system. The NameIdentifier claim, instead, is designed to hold a unique value, whether a GUID, a string or an integer. If you switch to User instead of NameIdentifier, make sure any value assigned to Name is unique per user.

All connections coming from an account with a matching name-identifier will be grouped together and will automatically be notified when the User client proxy is used. To switch to using the canonical Name claim, you need a custom IUserIdProvider, as follows:

```
public class MyUserIdProvider : IUserIdProvider
{
    public string GetUserId(HubConnectionContext connection)
    {
        return connection.User.Identity.Name;
    }
}
```

Needless to say, this component must be registered with the DI infrastructure during the startup phase. Here's the code to include in the ConfigureServices method of the startup class:

```
services.AddSignalR();
services.AddSingleton(typeof(IUserIdProvider), typeof(MyUserIdProvider));
```

At this point, everything is set up to have all matching user windows synchronized on the same state and view. How about balloon-style notifications?

The Final Touch

Adding and removing friends cause the need of refresh notifications being sent to the index page the current user is viewing. If a given user has two browser windows opened on different pages

of the same application (index and another page), she will receive refresh notifications only for the index page. However, adding and removing friends also causes add and remove notifications to be sent to the users that have been added or removed from the friend relationship list. For example, if user Dino decides to remove user Mary from his list of friends, user Mary will also receive a Remove notification. Ideally, a Remove (or Add) notification should reach the user regardless of the page being viewed, whether index or any other.

To achieve this, there are two options:

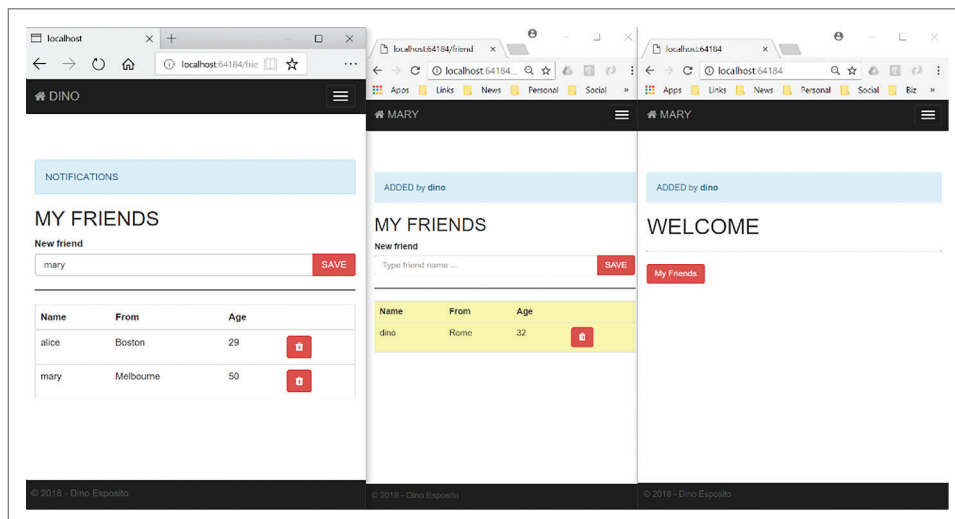


Figure 5 Cross-User Notifications

- Use a single SignalR hub with the connection setup moved to the layout page and then inherited by all pages based on that layout.
- Use two distinct hubs—one for refreshing the UI after adding or removing friends, and one to notify added or removed users.

If you decide to go with distinct hubs, the add/remove notification hub must be set up in all the pages where you want notifications to appear—most likely the layout pages you have in the application.

The sample application uses a single hub completely set up in the layout page. Note that the JavaScript object that references the current connection is globally shared within the client, meaning that the SignalR initialization code is better placed at the top of the layout body and before the Razor's RenderBody section.

Let's look at the final code of the Add method in the controller. This method is where the form in **Figure 1** ultimately posts. The method makes any necessary changes in the back-end to save friend relationships and then issues two SignalR messages—one to visually refresh the list of friends of the current user that did the operation, and a second to notify the added (or removed) user. The user is actually notified if she's currently connected to the application and from a page configured to receive and handle those specific notifications. **Figure 4** shows this.

In the layout page, some JavaScript code displays a balloon-style notification (or whatever type of UI adjustment you want to make). In the sample application, the notification takes the form of a message displayed on a notification bar for up to 10 seconds, as indicated in the code here:

```
friendConnection.on("added", (user) => {
    $("#notifications").html("ADDED by <b>" + user + "</b>");

    window.setTimeout(function() {
        $("#notifications").html("NOTIFICATIONS");
    },
    10000);
});
```

The results are shown in **Figure 5**.

A Word on SignalR Groups

In this article, I covered the SignalR support for selected notifications sent to a related group of users. SignalR offers a couple of approaches—the User client proxy and groups. The difference is subtle: The User client proxy implicitly generates a number of groups where the name is determined by the user ID and members are all connections opened by the same application user. A group is a more general mechanism that programmatically appends connections to a logical group. Both connections and name of the group are set programmatically.

Balloon-style notifications could have been implemented in both these approaches, but for this particular scenario the User client proxy was the most natural solution. Source code can be found at bit.ly/2HVYlp5. ■

DINO ESPOSITO has authored more than 20 books and 1,000 articles in his 25-year career. Author of "The Sabbatical Break," a theatrical-style show, Esposito is busy writing software for a greener world as the digital strategist at BaxEnergy. Follow him on Twitter: @despos.

msdnmagazine.com



Automate and add interactivity to your PDF applications

.NET IMAGING and PDF

- ☒ Interactive form field
- ☒ JavaScript actions
- ☒ Barcode and Signature field
- ☒ Annotation and Content editing



Professional SDK for building document management apps

- Image Viewer for .NET, WPF and WEB
- 100+ Image Processing and Document Cleanup commands
- PDF Reader, Writer, Visual Editor
- Image Annotations
- JBIG2 and JPEG2000 codecs
- OCR and Document Recognition
- Forms Processing and OMR
- DICOM decoder
- Barcode Reader and Generator
- TWAIN scanning

Free evaluation version
Royalty free licensing

www.vintasoft.com

VintaSoft® is a registered trademark
of VintaSoft Ltd.



JOIN US AT
The Ultimate Education Destination

2018 Orlando

ROYAL PACIFIC RESORT AT UNIVERSAL ORLANDO
DECEMBER 2-7, 2018



Live! 360: A Unique Conference for the IT and Developer Community

- 6 FULL Days of Training Including Hands-On Labs & Workshops
- 6 Co-Located Conferences for 1 Low Price
- Customize Your Own Agenda from Hundreds of Sessions
- Expert Education and Training
- Knowledge Share and Networking

CONNECT WITH LIVE! 360



twitter.com/live360
@live360



facebook.com
Search "Live 360"



linkedin.com
Join the "Live! 360" group!



instagram.com
@live360_events

EVENT PARTNERS



SILVER SPONSOR



SUPPORTED BY



AGENDAS JUST ANNOUNCED!

Check out live360events.com for full details.



SUMMER SAVINGS = BEST SAVINGS!

REGISTER NOW

REGISTER BY 8/31 AND SAVE \$500!

Use promo code: MSDN

See website for details.

6 CO-LOCATED CONFERENCES, 1 GREAT PRICE!

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

Visual Studio Live! features unbiased and practical development training on the Microsoft Platform. Come join us and code in paradise!

SQL Server LIVE!
TRAINING FOR DBAs AND IT PROS

SQL Server Live! will leave you with the skills needed to drive your data to succeed, whether you are a DBA, developer, IT Pro, or Analyst.

TECHMENTOR
IN-DEPTH TRAINING FOR IT PROS

TechMentor is where IT training meets sunshine, with zero marketing speak on topics you need training on now, and solid coverage on what's around the corner.

NEW! IN 2018 **Artificial Intelligence LIVE!**
AI FOR DEVELOPERS AND DATA SCIENTISTS

Artificial Intelligence Live! is an innovative, new conference for current and aspiring developers, data scientists, and data engineers covering artificial intelligence (AI), machine learning, data science, Big Data analytics, IoT & streaming analytics, bots, and more.

Office & SharePoint LIVE!
ON-PREMISE, CLOUD & CROSS-PLATFORM TRAINING

Office & SharePoint Live! provides leading-edge knowledge and training to work through your most pressing projects and enable people to work from anywhere at any time.

ModernApps LIVE!
MOBILE, CROSS-DEVICE & CLOUD DEVELOPMENT

Modern Apps Live!, presented in partnership with Magenic, focuses on how to architect, design and build a complete Modern App from start to finish.



LIVE360EVENTS.COM

VIRTUALIZATION
& Cloud Review

Visual Studio
MAGAZINE

PRODUCED BY
IIOS MEDIA
YOUR GROWTH, OUR BUSINESS.



Introduction to Q-Learning Using C#

Reinforcement learning (RL) is a branch of machine learning that tackles problems where there's no explicit training data with known, correct output values. Q-learning is an algorithm that can be used to solve some types of RL problems. In this article, I explain how Q-learning works and provide an example program.

The best way to see where this article is headed is to take a look at the simple maze in **Figure 1** and the associated demo program in **Figure 2**. The 3x4 maze has 12 cells, numbered from 0 to 11. The goal is to get from cell 8 in the lower-left corner, to cell 11 in the lower-right corner, in the fewest moves. You can move left, right, up or down, but not diagonally.

The demo program sets up a representation of the maze in memory and then uses the Q-learning algorithm to find a Q matrix. The Q stands for quality, where larger values are better. The row indices are the "from" cells and the column indices are the "to" cells. If the starting cell is 8, then scanning that row shows the largest Q value is 0.44 at to-cell 9. Then from cell 9, the largest value in the row is 1.08 at to-cell 5. The process continues until the program reaches the goal state at cell 11.

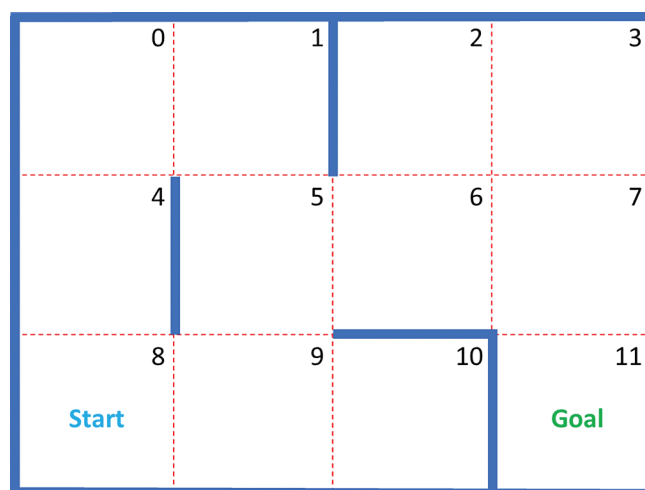
It's not likely you'll need to write code that solves a maze, but this is the Hello World for Q-learning because the problem is easy to grasp. I'll explain how Q-learning can generalize to more realistic problems later in this article.

This article assumes you have intermediate or better programming skills, but doesn't assume you know anything about Q-learning. The demo program is coded using C#, but you shouldn't have too much trouble refactoring the code to another language, such as Visual Basic or Python. The code for the demo program is presented in its entirety in this article and is also available in the accompanying file download.

Show Me the Code

For me at least, Q-learning is a bit unusual because I think the concepts are best understood by examining specific demo code rather than by starting with general principles. The overall structure of the demo program, with a few minor edits to save space, is shown in **Figure 3**.

To create the demo program, I launched Visual Studio and created a new C# console application project named QLearning. I used Visual Studio 2017, but the demo has no significant .NET dependencies so any version of Visual Studio will work fine. After the



$$Q(s_t, a_t) = [(1 - \alpha) * Q(s_t, a_t)] + [\alpha * (r_t + \gamma * \max_a Q(s_{t+1}, a) \forall a)]$$

Figure 1 Simple Maze Problem

template code loaded into the editor I removed all unneeded using statements, leaving just the reference to the System namespace. Then I added a reference to the Collections.Generic namespace because the demo uses a List<int> collection.

The demo program has a class-scope Random object because Q-learning involves a random selection component, as you'll see shortly. Variable ns stands for the number of states, which is synonymous with the number of cells in the maze. Object FT (feasible transitions) is an array-of-arrays-style matrix. Matrix R is the reward matrix, and matrix Q is the quality matrix.

The Q-learning algorithm requires parameters gamma (also known as the discount factor) and learnRate. I'll explain these later. Q-learning is iterative, so the demo sets up a maxEpochs variable to control how long the algorithm can use to find the Q matrix.

Setting up the Maze and the Rewards

The maze is created by method CreateMaze, which is defined as follows:

```
static int[][] CreateMaze(int ns) {
    int[][] FT = new int[ns][];
    for (int i = 0; i < ns; ++i) FT[i] = new int[ns];
    FT[0][1] = FT[0][4] = FT[1][0] = FT[1][5] = FT[2][3] = 1;
    FT[2][6] = FT[3][2] = FT[3][7] = FT[4][0] = FT[4][8] = 1;
    FT[5][1] = FT[5][6] = FT[5][9] = FT[6][2] = FT[6][5] = 1;
    FT[6][7] = FT[7][3] = FT[7][6] = FT[7][11] = FT[8][4] = 1;
    FT[8][9] = FT[9][5] = FT[9][8] = FT[9][10] = FT[10][9] = 1;
    FT[11][11] = 1; // Goal
    return FT;
}
```

Code download available at msdn.com/magazine/0818magcode.

Spreadsheets Everywhere.



SpreadsheetGear 2017 Released

SpreadsheetGear 2017 adds a new SpreadsheetGear for .NET Standard product, official support for Excel 2013 and Excel 2016, 51 new Excel functions for a total of 449 fully supported functions, full conditional formatting support, enhanced workbook protection and encryption, cell gradient rendering and more.



Support for iOS, Android, Linux, macOS, UWP and more

SpreadsheetGear for .NET Standard enables cross-platform developers to enjoy the same high performance Excel-compatible reporting, charting, calculations and more relied on by thousands of Windows developers for 10+ years.



Scalable Reporting

Easily create richly formatted Excel reports without Excel from any ASP.NET, Windows Forms, WPF or Silverlight application using spreadsheet technology built from the ground up for performance, scalability and reliability.



Windows
Forms



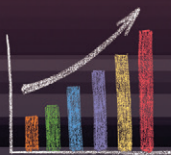
Silverlight



WPF

Powerful Controls

Add powerful Excel-compatible viewing, editing, formatting, calculating, filtering, sorting, charting, printing and more to your WinForms, WPF and Silverlight applications.



Comprehensive Charting

Enable users to visualize data with comprehensive Excel-compatible charting which makes creating, modifying, rendering and interacting with complex charts easier than ever before.



Fastest Calculations

Evaluate complex Excel-based models and business rules with the fastest and most complete Excel-compatible calculation engine available.

Download your free fully functional evaluation at SpreadsheetGear.com



SpreadsheetGear

Toll Free USA (888) 774-3273 | Phone (913) 390-4797 | sales@spreadsheetgear.com

```

file:///C:/QLearning/bin/Debug/QLearning.EXE
Begin maze RL Q-learning demo
Setting up maze and rewards
Analyzing maze using Q-learning
Done. Q matrix:

[ 0] 0.00 0.44 0.00 0.00 -0.04 0.00 0.00 0.00 0.00 0.00 0.00 0.00
[ 1] 0.12 0.00 0.00 0.00 0.00 0.00 1.08 0.00 0.00 0.00 0.00 0.00
[ 2] 0.00 0.00 0.00 0.00 2.35 0.00 0.00 0.00 2.35 0.00 0.00 0.00
[ 3] 0.00 0.00 0.00 1.08 0.00 0.00 0.00 0.00 4.90 0.00 0.00 0.00
[ 4] 0.12 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.12 0.00 0.00
[ 5] 0.00 0.44 0.00 0.00 0.00 0.00 0.00 2.35 0.00 0.00 0.44 0.00
[ 6] 0.00 0.00 0.00 1.08 0.00 0.00 1.08 0.00 4.90 0.00 0.00 0.00
[ 7] 0.00 0.00 0.00 0.00 2.35 0.00 0.00 2.35 0.00 0.00 0.00 10.00
[ 8] 0.00 0.00 0.00 0.00 -0.04 0.00 0.00 0.00 0.00 0.44 0.00 0.00
[ 9] 0.00 0.00 0.00 0.00 0.00 0.00 1.08 0.00 0.00 0.12 0.00 0.00
[10] 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.44 0.00 0.00
[11] 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

Using Q to walk from cell 8 to 11
8->9->5->6->7->11->done

End demo

```

Figure 2 Q-Learning Demo Program

The method returns a matrix that defines allowable moves. For example, you can move from cell 4 to cell 8, but you can't move from cell 4 to cell 5 because there's a wall in the way. Recall that C# initializes int arrays to 0, so CreateMaze needs to specify only allowable moves. Notice that you can't move from a cell to itself, except for the goal-cell 11.

The reward matrix is defined by:

```

static double[][] CreateReward(int ns) {
    double[][] R = new double[ns][];
    for (int i = 0; i < ns; ++i) R[i] = new double[ns];
    R[0][1] = R[0][4] = R[1][0] = R[1][5] = R[2][3] = -0.1;
    R[2][6] = R[3][2] = R[3][7] = R[4][0] = R[4][8] = -0.1;
    R[5][1] = R[5][6] = R[5][9] = R[6][2] = R[6][5] = -0.1;
    R[6][7] = R[7][3] = R[7][6] = R[7][11] = R[8][4] = -0.1;
    R[8][9] = R[9][5] = R[9][8] = R[9][10] = R[10][9] = -0.1;
    R[7][11] = 10.0; // Goal
    return R;
}

```

In this example, moving to goal-cell 11 gives a reward of 10.0, but any other move gives a negative reward of -0.1. These values are somewhat arbitrary. In general, when working with RL, the reward structure is entirely problem-dependent. Here, the small negative reward punishes every move a bit, which has the effect of preferring shorter paths over longer paths to the goal. Notice you don't have to set a reward for moves that aren't allowed because they will never happen.

The goal of Q-learning is to find the value of the Q matrix. Initially, all Q values are set to 0.0 and the Q matrix is created like so:

```

static double[][] CreateQuality(int ns) {
    double[][] Q = new double[ns][];
    for (int i = 0; i < ns; ++i)
        Q[i] = new double[ns];
    return Q;
}

```

Defining Possible Next States

As you'll see shortly, the Q-learning algorithm needs to know what states the system can transition to, given a current state. In this example, a state of the system is the same as the location in the maze so there are only 12 states. Method GetPossNextStates is defined like so:

```

static List<int> GetPossNextStates(int
s, int[][] FT) {
    List<int> result = new List<int>();
    for (int j = 0; j < FT.Length; ++j)
        if (FT[s][j] == 1) result.Add(j);
    return result;
}

```

For example, if the current state *s* is 5, then GetPossNextStates returns a List<int> collection holding (1, 6, 9). The Q-learning algorithm sometimes goes from the current state to a random next state. That functionality is defined by method GetRandNextState:

```

static int GetRandNextState(int s,
int[][] FT) {
    List<int> possNextStates =
        GetPossNextStates(s, FT);
    int ct = possNextStates.Count;
    int idx = rnd.Next(0, ct);
    return possNextStates[idx];
}

```

So, if the current state *s* is 5, then GetRandNextState returns either 1 or 6 or 9 with equal probability (0.33 each).

The Q-Learning Algorithm

The key update equation for Q-learning is based on the mathematical Bellman equation and is shown at the bottom of Figure 1. The algorithm is implemented in method Train. In high-level pseudo-code, the Q-learning algorithm is:

```

loop maxEpochs times
    set currState = a random state
    while currState != goalState
        pick a random next-state but don't move yet
        find largest Q for all next-next-states
        update Q[currState][nextState] using Bellman
        move to nextState
    end-while
end-loop

```

The algorithm is not at all obvious, and for me, it's best understood by examining the code. The definition begins:

```

static void Train(int[][] FT, double[][] R, double[][] Q,
int goal, double gamma, double lrnRate, int maxEpochs)
{
    for (int epoch = 0; epoch < maxEpochs; ++epoch) {
        int currState = rnd.Next(0, R.Length);
        ...
    }
}

```

The number of training epochs must be determined by trial and error. An alternative design is to iterate until the values in the Q matrix don't change, or until they stabilize to very small changes per iteration. The inner loop iterates until the current state becomes the goal state, cell 11 in the case of the demo maze:

```

while (true) {
    int nextState = GetRandNextState(currState, FT);
    List<int> possNextNextStates = GetPossNextStates(nextState, FT);
    double maxQ = double.MinValue;
    for (int j = 0; j < possNextNextStates.Count; ++j) {
        int nns = possNextNextStates[j]; // short alias
        double q = Q[nextState][nns];
        if (q > maxQ) maxQ = q;
    }
}

```

Imagine you're in a maze. You see that you can go to three different rooms, A, B, C. You pick B, but don't move yet. You ask a friend to go into room B and the friend tells you that from room B you



TECH EVENTS WITH PERSPECTIVE



TECHMENTOR

IN-DEPTH TRAINING FOR IT PROS

ROYAL PACIFIC RESORT AT
UNIVERSAL ORLANDO
December 2-7, 2018



Where IT Training Meets the Sunshine

TechMentor offers in-depth training for IT Pros, giving you the perfect balance of the tools you need today, while preparing you for tomorrow. Expect troubleshooting tips, performance optimization training, and best practices from peers and experts in the industry. Plus, there will be dedicated coverage of Windows PowerShell, core Windows Server functionality, Security, System Center and so much more. We'll see you in the sun!

TRACK TOPICS INCLUDE:

- Client and Endpoint Management
- PowerShell and DevOps
- Infrastructure
- Soft Skills for ITPros
- Security and Ethical Hacking
- Cloud (Public/Hybrid/Private)

REGISTER
NOW

SAVE \$500 WITH SUMMER SAVINGS!
REGISTER BY AUGUST 31

Use Promo Code MSDN

A Part of Live! 360: The Ultimate Education Destination

6 CONFERENCES, 1 GREAT PRICE

Visual Studio

SQL Server

TECHMENTOR

Artificial Intelligence

Office & SharePoint

Modern Apps



TECHMENTOREVENTS.COM/ORLANDO

EVENT PARTNERS



SILVER SPONSOR



PREMIER & ALLIANCE MEDIA PARTNERS



SUPPORTED BY



PRODUCED BY



can go to rooms X, Y, Z and that of those rooms Y has the best Q value. In other words, Y is the best next-next state.

The update to the Q matrix is performed:

```
...
    Q[currState][nextState] =
        ((1 - lrnRate) * Q[currState][nextState]) +
        (lrnRate * (R[currState][nextState] + (gamma * maxQ)));

    currState = nextState;
    if (currState == goal) break;
} // while
} // for
} // Train
```

The value of gamma, the discount factor, influences the importance of future rewards.

The update equation has two parts. The first part, $((1 - \text{lrnRate}) * Q[\text{currState}][\text{nextState}])$, is called the exploit component and adds a fraction of the old value. The second part, $(\text{lrnRate} * (R[\text{currState}][\text{nextState}] + (\text{gamma} * \text{maxQ})))$, is called the explore component. Larger values of the `lrnRate` increase the influence of both current rewards and future

Figure 3 Q-Learning Demo Program Structure

```
using System;
using System.Collections.Generic;
namespace QLearning
{
    class QLearningProgram
    {
        static Random rnd = new Random(1);

        static void Main(string[] args)
        {
            Console.WriteLine("Begin Q-learning maze demo");
            Console.WriteLine("Setting up maze and rewards");
            int ns = 12;
            int[][] FT = CreateMaze(ns);
            double[][] R = CreateReward(ns);
            double[][] Q = CreateQuality(ns);

            Console.WriteLine("Analyzing maze using Q-learning");
            int goal = 11;
            double gamma = 0.5;
            double learnRate = 0.5;
            int maxEpochs = 1000;
            Train(FT, R, Q, goal, gamma, learnRate, maxEpochs);
            Console.WriteLine("Done. Q matrix: ");
            Print(Q);
            Console.WriteLine("Using Q to walk from cell 8 to 11");
            Walk(8, 11, Q);
            Console.WriteLine("End demo");
            Console.ReadLine();
        }

        static void Print(double[][] Q) { . . . }
        static int[][] CreateMaze(int ns) { . . . }
        static double[][] CreateReward(int ns) { . . . }
        static double[][] CreateQuality(int ns) { . . . }
        static List<int> GetPossNextStates(int s,
            int[][] FT) { . . . }
        static int GetRandNextState(int s, int[][] FT) { . . . }
        static void Train(int[][] FT, double[][] R, double[][] Q,
            int goal, double gamma, double lrnRate,
            int maxEpochs) { . . . }
        static void Walk(int start, int goal, double[][] Q) { . . . }
        static int ArgMax(double[] vector) { . . . }
    } // Program
} // ns
```

rewards (explore) at the expense of past rewards (exploit). The value of gamma, the discount factor, influences the importance of future rewards.

Using the Q Matrix

After the quality matrix has been computed, it can be used to find an optimal path from any starting state to the goal state. Method `Walk` implements this functionality:

```
static void Walk(int start, int goal, double[][] Q) {
    int curr = start; int next;
    Console.Write(curr + "->");
    while (curr != goal) {
        next = ArgMax(Q[curr]);
        Console.Write(next + "->");
        curr = next;
    }
    Console.WriteLine("done");
}
```

Notice the method assumes that the goal state is reachable from the starting state. The method uses helper `ArgMax` to find the best next state:

```
static int ArgMax(double[] vector) {
    double maxVal = vector[0]; int idx = 0;
    for (int i = 0; i < vector.Length; ++i) {
        if (vector[i] > maxVal) {
            maxVal = vector[i]; idx = i;
        }
    }
    return idx;
}
```

For example, if a vector has values (0.5, 0.3, 0.7, 0.2) then `ArgMax` returns 2. The demo defines a `Print` method to display the Q matrix. You can get the pretty-print version in the accompanying file download. A simplified version is:

```
static void Print(double[][] Q) {
    int ns = Q.Length;
    Console.WriteLine("[0] [1] . . [11]");
    for (int i = 0; i < ns; ++i) {
        for (int j = 0; j < ns; ++j) {
            Console.Write(Q[i][j].ToString("F2") + " ");
        }
        Console.WriteLine();
    }
}
```

Wrapping Up

The Q-learning example presented here should give you a good understanding of the main principles involved. The problem scenario presented in this article is one with discrete states, but Q-learning can work with continuous states, too. The general challenge is to maximize the long-term reward, which for the maze example is the same as getting to the goal state in the fewest number of moves. Q-learning can be useful when you can safely train a system with many trials, such as training an industrial robot how to perform a task. But Q-learning isn't applicable in scenarios like training a driverless car how to navigate through traffic. Q-learning and RL remind me somewhat of neural networks in the 1980s—there are relatively few practical applications right now, but there are intense research efforts. Many of my colleagues believe that at some point RL will explode into usefulness in unexpected ways. ■

DR. JAMES MCCAFFREY works for Microsoft Research in Redmond, Wash. He has worked on several Microsoft products, including Internet Explorer and Bing. Dr. McCaffrey can be reached at jamccaff@microsoft.com.

THANKS to the following Microsoft technical experts who reviewed this article: Asli Celikyilmaz, Chris Lee, Ricky Loynd, Amr Sharaf, Ken Tran



SQL Server® LIVE!

TRAINING FOR DBAs AND IT PROS

ROYAL PACIFIC RESORT AT
UNIVERSAL ORLANDO
December 2-7, 2018



Data. Driven.

With timely, relevant content delivered by recognized experts, who, like you, are driven by data, SQL Server Live! is back to help you do more with your SQL Server investment. With 6 days of workshops, deep dives and breakout sessions, SQL Server Live! provides a broad range of sessions on everything from performance tuning to security, to reporting and data integration. ***This training event is for administrators, DBAs, developers and BI pros to improve old approaches, adopt new techniques, and to modernize the SQL Server infrastructure.***

TRACK TOPICS INCLUDE:

- BI, Big Data, Data Analytics, and Data Visualization
- SQL Server Administration & Maintenance
- SQL Server in the Cloud
- SQL Server for Developers
- SQL Server Performance Tuning and Optimization

REGISTER
NOW

SAVE \$500 WITH SUMMER SAVINGS!
REGISTER BY AUGUST 31

Use Promo Code MSDN

A Part of Live! 360: The Ultimate Education Destination

6 CONFERENCES, 1 GREAT PRICE

Visual Studio LIVE! | SQL Server LIVE! | **TECHMENTOR** | Artificial Intelligence LIVE! | Office & SharePoint LIVE! | Modern Apps LIVE!



SQLLIVE360.COM

EVENT PARTNERS



SILVER SPONSOR



PREMIER & ALLIANCE MEDIA PARTNERS



SUPPORTED BY



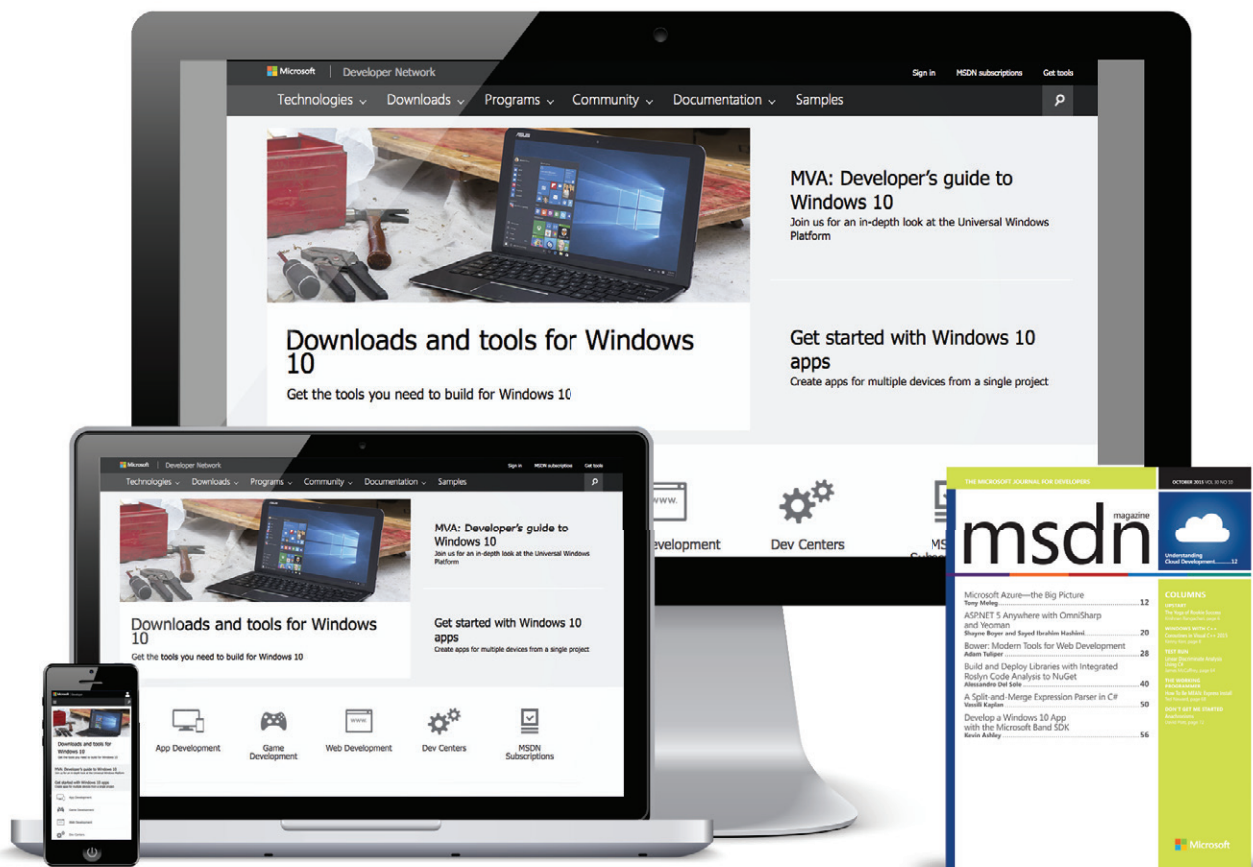
PRODUCED BY



msdn

magazine

Where you need us most.



Visual Studio **LIVE!**
EXPERT SOLUTIONS FOR .NET DEVELOPERS

LIVE!
360
TECH EVENTS WITH PERSPECTIVE

MSDN.microsoft.com



Visual Studio **LIVE!**

EXPERT SOLUTIONS FOR .NET DEVELOPERS

**ROYAL PACIFIC RESORT AT
UNIVERSAL ORLANDO
December 2-7, 2018**



Coding in Paradise

Developers, engineers, software architects and designers will code in paradise this December at Visual Studio Live! (VSLive!™). Help us celebrate 25 years of coding innovation by returning to warm, sunny Orlando for six days of unbiased and cutting-edge education on the Microsoft Platform. Soak in the knowledge on everything from Visual Studio and the .NET framework, to ASP.NET, .NET Core, JavaScript, Xamarin, Database Analytics, and so much more. VSLive!™ features 60+ sessions led by industry experts and Microsoft insiders.

Grab your flip flops, and your laptops, and make plans to attend the conference more developers rely on to expand their .NET skills and the ability to build better applications.

**REGISTER
NOW**

**SAVE \$500 WITH SUMMER SAVINGS!
REGISTER BY AUGUST 31**

Use Promo Code MSDN

A Part of Live! 360: The Ultimate Education Destination

6 CONFERENCES, 1 GREAT PRICE

Visual Studio **LIVE!** | SQL Server **LIVE!** | **TECHMENTOR** | Artificial Intelligence **LIVE!** | Office & SharePoint **LIVE!** | Modern Apps **LIVE!**



VSLIVE.COM/ORLANDO

EVENT PARTNERS



SILVER SPONSOR



PREMIER & ALLIANCE MEDIA PARTNERS



SUPPORTED BY



PRODUCED BY





Sing in Me

*Sing in me, Muse, and through me tell the story
of that man skilled in all ways of contending,
the wanderer, harried for years on end*

...

*Of these adventures, Muse, daughter of Zeus,
tell us in our time, lift the great song again.*

"The Odyssey," Homer (c. 750 BCE), Robert Fitzgerald translation (1961)

I've just returned from keynoting a conference in Greece, explaining "Why Software STILL Sucks." I love that country and its friendly, hospitable people. Still groaning under their own economic catastrophe, they risk their lives at sea to save drowning refugees (see nyti.ms/2K1jun9). I admire their contribution to Western civilization, and I'd like to share some of it with you. (Want me to write about your country? Invite me to your conference. I give great keynote talks.)

You have no doubt heard of the Muses (en.wikipedia.org/wiki/Muses). In classical Greek mythology, these nine goddesses are the source of inspiration in literature, science and the arts. They give us our English words for amusement, museum and even music. The smiling and frowning masks you see displayed in many theaters belong to Thalia, muse of comedy, and Melpomene, muse of tragedy. When you put on a CD or spin up Spotify, a nod to Euterpe (music) wouldn't be out of place, or a tip of your shoes to Terpsichore when you dance to it.

Authors and poets commonly invoke their muses at the start of a work, as Homer does in the beginning of "The Odyssey" (quoted above). Some of our best geek writers do so via rigid poetic forms. Seth Schoen, in his superb 456-stanza DeCSS haiku (bit.ly/2JQRzXE, 2001), invokes his muse thus:

*Now help me, Muse, for
I wish to tell a piece of
controversial math,*

*for which the lawyers
of DVD CCA
don't forbear to sue:*

*that they alone should
know or have the right to teach
these skills and these rules.*

Science fiction author Neal Stephenson opens his novel, "Quicksilver" (William Morrow, 2003), with a sonnet, from which I excerpt:

*State your intentions, Muse. I know you're there.
Dead bards who pined for you have said
You're bright as flame, but fickle as the air.*

...

Why rustle in the dark, when fledged with fire?

*Craze the night with flails of light. Reave
Your turbid shroud. Bestow what I require.*

Which muse inspires us software developers? Urania, muse of astronomy? Astronomy leads to navigation, and navigation to mathematics. She is often portrayed holding a compass. So Urania? Maybe. Calliope, muse of epic poetry, muse of Homer? She's often portrayed holding a writing tablet, obviously an early iPad prototype. So Calliope? Maybe. The last project I was called in too late to save contained elements of both comedy and tragedy, so the electron/positron duo of Thalia and Melpomene? Maybe.

But software development is new, different from anything humans have previously done. We need our own muse. And she appeared to me, in a jet-lagged vision, as dawn broke over the wine-dark Aegean Sea.

Her name is Monomidene (Greek: Μονομηδένη), from the Greek words for one and zero, the atomic components of all our invocations. The nine original muses were daughters of Zeus and Mnemosyne, the Titan goddess of memory. And goodness knows, our programs consume more memory every year. But see **Figure 1**. She's wearing the hat of a female senior officer in the U.S. Navy. Could Monomidene be the daughter of Zeus and Grace Hopper, a goddess in her own right? Scoff if you must, but I think Amazing Grace gave this industry more than just a COBOL compiler.

Each muse carries the tools of her trade, such as Terpsichore's lyre or Euterpe's panpipes. Monomidene carries a flyswatter, obviously for crushing bugs, like her mother's original in the Smithsonian Museum of American History (bit.ly/2yBHXLI). Also, a large mug of coffee, for all those late-night debugging sessions. What more could any programmer want?

Monomidene stands ready for your invocations. Tell me how she inspires you. ■

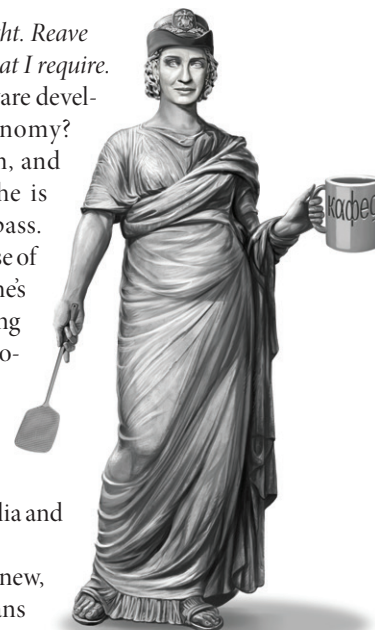


Figure 1
**Monomidene, Muse of
Software Development**

DAVID S. PLATT teaches programming .NET at Harvard University Extension School and at companies all over the world. He's the author of 11 programming books, including "Why Software Sucks" (Addison-Wesley Professional, 2006) and "Introducing Microsoft .NET" (Microsoft Press, 2002). Microsoft named him a Software Legend in 2002. He wonders whether he should tape down two of his daughter's fingers so she learns how to count in octal. You can contact him at rollthunder.com.

Empower your development. Build better applications.



GrapeCity's family of products provides **developers, designers, and architects** with the ultimate collection of easy-to-use tools for building **sleek, high-performing, feature-complete** applications. With over 25 years of experience, we understand your needs and offer the industry's best support. **Our team is your team.**



ComponentOne
.NET UI CONTROLS



ActiveReports
REPORTING SOLUTIONS



Spread
SPREADSHEET SOLUTIONS



Wijmo
JAVASCRIPT UI CONTROLS



Documents
DOCUMENT APIS

For more information: **1-800-831-9006**

Learn more and get free 30-day trials at **GrapeCity.com**



Modern UI Made Easy



15% off on new purchases by August 31, 2018. See www.telerik.com/msdn for details.

Building a modern UI for Web, Desktop and Mobile apps has never been easier
with our .NET, JavaScript & Productivity Tools

www.telerik.com/msdn